DATEL Electronics

Limited

Amiga ACTION REPLAY MK III

FROM

EUROSYSTEMS

INSTRUCTION MANUAL

Inhaltsverzeichnis
==========

1. Handhabung der Bremse

Die Bremse, die Ihren Amiga nahezu stufenlos fast bis zum Stillstand abbremsen kann, kann zu jeder Zeit zu- und abgeschaltet werden. Den augenblicklichen Zustand der Bremse erkennen Sie an der roten Brems-Lampe. Leuchtet diese, so ist die Bremse aktiv! Mit dem runden Drehregler können Sie die Geschwindigkeit, mit der Ihr Amigaprogramm arbeitet, stufenlos herunterregeln.
Die Bremse wird automatisch, während Sie im Modul-Editor arbeiten, ausgeschaltet, so daß sämtliche Cartridge-Befehle immer mit maximaler Geschwindigkeit abgearbeitet werden.

ACHTUNG: Bei manchen Programmen ist es möglich, daß Sie mit eingeschalteter Bremse nicht mehr einwandfrei laufen. Man sollte dann die Bremse abschalten!

2. Befehlseingabe

Haben Sie den Modul-Editor Taster gedrückt, erscheint der Editor-Bildschirm mit weißer Schrift auf blauem Grund (falls nicht von Ihnen geändert mit dem COLOR-Befehl oder der Preferences-Seite). Außerdem erscheint ein blinkender Cursor (weißer Unterstrich "_"). Sie können jetzt mit der Befehlseingabe beginnen!
Falls Sie einmal einen Befehl vergessen haben, betätigen Sie einfach die Help-Taste. Es wird dann eine Liste aller Befehle und Editor-Tasten ausgegeben - zur Gedächtnisstütze.
Zur Befehlseingabe stehen Ihnen folgende Editor-Tasten zur Verfügung:

F1	löscht den aktuellen Bildschirminhalt
<SHIFT> F1	setzt den CURSOR in die linke obere Bildschirmecke
F2	Kopiert die zweite Bildschirmseite auf die aktuelle Bildschirmseite
<SHIFT> F2	Kopiert die aktuelle Bildschirmseite auf die zweite Bildschirmseite
F3	Preferences-Menü siehe Preferences
F4	wiederholt den zuletzt eingegebenen Befehl
F5	gibt den gesammten Modul-Bildschirm über einen evtl. angeschlossenen EPSON-kompatiblen Drucker aus
F6	schaltet das Printerprotokoll aller Befehls Ein- und Ausgaben ein/aus
F7	wählt zwischen den Modi Überschreiben und Einfügen bei der Befehlseingabe
F8	gibt die Tastaturbelegung/Sonderfunktionen während des P-Befehls aus (MemoryPeeker)
F9	schaltet zwischen deutschem und amerikanischem Tastaturtreiber um
F10	wechselt den Eingabe-Bildschirm mit dem zweiten Eingabe-Bildschirm
HELP	gibt einen Hilfstext aus

<SHIFT>HELP	gibt eine Liste der abgekürzbaren Befehle aus
SHIFT	stoppt das Bildschirmrollen, so daß Sie sich alles in Ruhe ansehen können
TAB	Fügt ein Leerzeichen ein
DEL	Entfernt das Zeichen unter dem CURSOR
BACKSPACE	Entfernt das Zeichen vor dem CURSOR
LINKER MAUSKNOPF	bricht Druckerausgaben des Moduls ab

Die vier CURSOR-Tasten werden auch unterstützt. Betätigt man diese mit der SHIFT-Taste, erfüllen Sie auch Sonderfunktionen, wie in erste/letzte Zeile springen und zum nächsten/vorigen Wort springen.

WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG
Viele Befehle wie z.B. der Speicherdump, der Disassembler, ASCII-Ausgabe u.a., bieten folgende komfortable Möglichkeit: Sind einmal Informationen (z.B. Speicherinhalte oder beim Disassembler Maschinenbefehle) oben aus dem Bildschirm herausgerollt, so können diese durch die Cursor-hoch-Taste wieder nach unten gerollt werden,falls der Cursor sich in der obersten Bilschirm-Zeile befindet. Die jeweiligen Adressen werden dann entsprechend erniedrigt!!

Bitte drucken Sie nun einmal den Modul-Editor Taster und probieren Sie die Editortasten aus.

Beachten Sie, daß jede Befehlszeile mit einem RETURN abgeschlossen sein muß.
Während der Befehlsabarbeitung blinkt die Power LED des Amigas und zeigt damit an, daß er beschäftigt ist. Fast alle Befehle lassen sich mit der ESC-Taste abbrechen, Druckerausgaben mit der linken Maustaste!

3. Zahleneingabe

Beim ACTION REPLAY AMIGA können Sie sämtliche Zahleneingaben bei Befehlen hexadezimal, dezimal oder binär angeben. Außerdem können anstelle von Zahlen Registersymbole verwendet werden, mit denen man den aktuellen Inhalt der Prozessor-Register erhält.
Dabei interpretiert das Modul "normal", d.h. ohne besondere Kennzeichnung, eingegebene Zahlen als Hexadezimalzahlen.
D.h. wenn Sie "12" eingeben interpretiert dies das Modul als Hexadezimalzahl mit dem Wert (dezimal) !18. Um ausdrücklich anzugeben, daß es sich bei einer eingegebenen Zahl um eine Hexadezimalzahl handelt, kann der Zahl auch ein Dollarzeichen "$" vorangestellt werden: $12.
Wollen Sie eine Zahl dezimal interpretiert wissen, muß diese durch ein vorangestelltes Ausrufungszeichen "!" gekennzeich-

net werden, z. B. "!32" oder "!65535". Binäre Zahlen dagegen markieren Sie mit einem vorangestellten Prozentzeichen "%", z.B. "%1001" oder "%00111101011".
Registerangaben müssen durch ein vorangestelltes Backslash "\" gekennzeichnet werden. Erlaubt sind hierbei folgende Bezeichnungen:
 \D0 ... \D7: Datenregister D0 bis D7
 \A0 ... \A7: Adressregister A0 bis A7
 \SP: Userstackpointer
 \PC: ProgramCounter
 \B: Dmon-Buffer start (s. auch DMON-Befehl)

Bei allen Zahlenangaben können anstatt einer Zahl auch mathematische Ausdrücke angegeben werden, die die Operationen plus, minus, mal und geteilt enthalten dürfen.

Bei Zahleneingaben werden also folgende Zahlen vom Modul als gleich angesehen:
      !128 = $80 = 8*!16 = $40+40 = 80 = %10000000
      %001001010 = 4A - 94/2-10+!16 = !74 = $4A

Führende Nullen können bei der Adressen- und Zahleneingaben weggelassen werden!
z.B.: $00052340 = 52340

Wollen Sie negative Zahlen eingeben, können Sie dies durch voranstellen eines Minus-zeichens bei allen Zahlenformaten erreichen.
z.B.: -$2 = -2 = -!2 = -%10 = 1-3 (= $fffffffe)

Hexadezimalsystem:
Das Hexadezimale Zahlensystem stellt Zahlen zur Basis 16 dar, während im Dezimalsystem mit der Basis 10 gerechnet wird. Im Hexadezimalsystem sind daher Ziffern für die Zahlen 11-15 nötig. Es werden dafür die Ziffern "A" bis "F" verwendet.
Es gilt folgende Übersetzungstabelle:
A = !10, B = !11, C = !12, D = !13, E = !14, F = !15

Beispiel:
hex to dez: $FE = 15*16^1 + 14*16*0 = !254
            $3E8 = 3*16^2 + 14*16^1 + 8*16^0 = !1000

Der Calculate Befehl nimmt automatisch alle notwendigen Umwandlungen vor und gibt das Ergebnis in allen Zahlensystemen aus, wobei die dezimale Ausgabe unter Berücksichtigung des Vorzeichens erfolgt!
z.B.: "? $1000 - $100 + !256 - %1 = $fff"

4. Die Preferences-Seiten

Wenn Sie im Modul-Editor anstatt einen Befehl einzugeben die

Taste F3 drücken kommen Sie zur ersten Preferences-Seite. Mit der linken Maustaste können Sie nun die einzelnen Felder (=Gadgets) anklicken. Statt der Maus können Sie aber auch einen Joystick im Port 1 benutzen. Ein aktiviertes Feld wird durch eine inverse Farbe dargestellt. Verlassen können Sie die Preferences-Seiten durch anklicken von "USE" oder durch drücken der ESC-Taste es werden aber auf jeden Fall die geänderten Einstellungen übernommen
Um zwischen den zwei Preferences-Seiten umzuschalten, können Sie entweder das entsprechende Feld anklicken oder die Leertaste (SPACE) betätigen.

Nun die Bedeutung der anklickbaren Felder im einzelnen.

4.1 Erste Preferences-Seite:

4.1.1 Links Oben: MEMORYCONTROLL

Durch Anklicken von den FAST Gadgets können Sie ihr Fastmem ,falls vorhanden, an- und abschalten. Mit dem CHIP Gadget können Sie, falls möglich, zwischen 512K und dem maximalen (z.Zt. 1Mb) Chipmem wählen. Durch das Anwählen des CLEAR RAM Gadget wird das Speicherlöschen beim Reset eingeschaltet.
Die geänderte Speicherkofiguration wird aber erst beim nächsten Reset aktiv.
Für die Benutzer einer externen Fast-RAM Erweiterung sind folgende Felder interessant:
Unterhalb von "Extern Memory" werden die vom Modul erkannten belegten Bereiche externer Speichererweiterungen ausgegeben. Dies funktioniert automatisch bei autokonfigurierenden Speichererweiterungen. Diese Werte können aber auch durch anklicken verändert werden, um z.B. nicht autokonfigurierende Speichererweiterungen dem Modul bekanntzumachen. (Nach anklicken zuerst die DEL-Taste betätigen, um den alten Wert zu löschen!)
Das Add-Feld dient dazu Speichererweiterungen bei jedem RESET automatisch in das Amiga-Betriebsystem einzubinden (ersetzt Addmem-Befehle). Dieses Feld darf nicht benutzt werden, wenn die von Ihnen benutzte Speichererweiterung autokonfigurierend ist, es sei denn, Sie haben die Autokonfiguration mit Hilfe des AUTOCONFIG-Feldes (s.dort) ausgeschaltet.
Außerdem können Sie mit Hilfe dieser Felder nur Teile Ihrer Speichererweiterung in das Amiga-Betriebssystem einbinden, so daß die verbleibenden freien Bereiche z.B. für Modul-Funktionen zur Verfügung stehen (siehe SQMEM-Befehl, Modul Interna-Felder).
Bsp.: 1. Sie besitzen eine autokonfigurierende 2MB Speichererweiterung:
        Einschalten der vollen 2Mb Erweiterung:
          AUTOCONFIG-Feld auf ON, Add-Feld auf OFF
        Ausschalten der 2Mb Erweiterung:
          AUTOCONFIG-Feld auf OFF, Add-Feld auf OFF
        Einschalten eines Teils der Erweiterung (z.B. 512kB):
          Eintragen des gewünschten Bereiches bei Extern

Memory (muß innerhalb von $200000-$400000 liegen; im Bsp: $200000-$280000), AUTOCONFIG-Feld auf OFF, Add-Feld auf ON

2. Sie besitzen eine nicht autokonfigurierende 2Mb Speichererweiterung:
   Einschalten von Teilen oder der vollen 2Mb Erweiterung:
   Eintragen der Lage des einzubindenden Speicherbereichs (s.o.), Add-Feld auf ON
   Ausschalten der Erweiterung:
   Add-Feld auf OFF
3. Sie besitzen eine autokonfigurierende 4Mb Erweiterung:
   Einschalten der vollen 4Mb Erweiterung:
   AUTOCONFIG-Feld auf OFF, Add-Feld auf ON
   Ausschalten der Erweiterung:
   AUTOCONFIG-Feld auf OFF, Add-Feld auf OFF
   Einbinden von Teilen s. 1.)

4.1.2 Links Mitte: MODULE INTERNA

NoRes-Feld: Sämtliche Aktivitäten des Moduls beim RESET werden unterlassen. So unterbleibt z.B das Speicherlöschen, aber auch Modul-interne Initialisierungen.
Test1-Feld und Test2-Feld: Einschalten der jeweiligen Modulinternen Einsprung Tests zur Erhöhung der Softwarekompatibilität. Wird das Test2-Feld aktiviert muß genügend freier, d.h. dem Amiga Betriebssystem unbekannter Speicher zur Verfügung stehen. Mindestens aber soviel, wie ChipMem aktiv ist (s.Memory Control Felder)
Blanker-Feld: schaltet den Bildschirmschoner des Modul-Editors an und aus.

4.1.3 Links Unten: COLOR CONTROL

Durch diese Gadgets können Sie ihre Farben im Modul selber einstellen.
Mit BACKGND und FOREGND und den Farbwerten können Sie die Farbe der Zeichen und des Hintergrundes frei wählen.

4.1.4 Mitte Oben: MEGASTICK

schaltet die Funktionen der Joystick-Key-Simulation an und aus (siehe auch MEGASTICK-Befehl)

4.1.5 Mitte Mitte: AUTOCONFIG

schaltet den Autokonfigurations-Prozess des Amiga-Betriebssystems an und aus.
Mit diesem Gadget können Sie ihre speicherweiterungen und Festplatten ausschalten sofern sie autokonfigurierend sind.

4.1.6 Mitte Unten: OK und NEXT PAGE

Beim Anklicken von OK wird die Preferences-Seite verlassen und die Einstellungen übernommen. (Gleiche Funktion wie ESC-Taste)
Beim Anklicken von NEXT PAGE wird auf die zweite Preferences-Seite umgeschaltet (Gleiche Funktion wie SPACE-Taste)

4.1.7 Rechts: AUTOFIRE

Beim Anklicken der verschiedenen Prozentmarken (0%-100%) kann für Joysticks oder Maus ein Dauerfeuer simuliert werden, wobei 100% für die größtmögliche Schußfrequenz und 0% für manuelles Bedienen des Feuerknopfes steht. Bei manchen Programmen kann aber die maximale Frequenz schon unter 100% liegen.

4.2 Zweite Preferences-Seite:

4.2.1 Links Oben: BOOTSELECTOR

Die BOOTSEL Felder bestimmen von welchem Laufwerk beim nächsten Reset gebootet werden soll.
OFF: Normales Booten von Laufwerk df0
VARIABLE: Booten von jedem Laufwerk indem eine Diskette bei Reset liegt die Priorität der Laufwerke beträgt dabei df3>df2>df1>df0
df0-df3: Booten von dem gewählten Laufwerk

4.2.2 Links Mitte: BOOTBLOCKCODER

ON-Feld: schaltet den Bootblockcoder ein (s. auch BOOTCODE-Befehl), dieser arbeitet nur bei gleichzeitig eingeschaltetem Bootvirustest (s.VIRUSTEST Boot-Feld)
Code-Feld: Dient zum Eintragen des aktuellen Bootcodes (s. auch BOOTCODE-Befehl; zum Eintragen mit DEL-Taste alten Code löschen)

4.2.3 Links Unten: DISKCODER

DF0-DF3 - Felder: einschalten des Diskcoders des jeweiligen Laufwerks (s. auch CODE-Befehl)
Code-Felder: Dienen zum Eintragen der jeweiligen Codewörter (s. auch CODE-Befehl; zum Eintragen mit DEL-Taste alten Code löschen)

4.2.4 Mitte Oben: DRIVECONTROL

Mit den ON/OFF Gadgets können Sie ihre Diskettenlaufwerke abschalten (Laufwerk df0 darf nicht bei Kick 1.2/1.3 abgeschaltet werden).

4.2.5 Mitte Mitte: VIRUSTEST

FIND-Feld: Falls aktiviert durchsucht das Modul bei jedem RESET und Modul-Aufruf mittles des Tasters den Speicher des Amigas nach Viren/resetfesten Programmen und signalisiert deren Fund (Beim Reset blinkt der Bildschirm; beim Aufruf des Moduls wird eine Meldung ausgegeben)
KILL-Feld: Falls aktiviert wird bei jedem RESET und Modul-Aufruf versucht, gefundene Viren/resetfeste Programme im Speicher zu entfernen (nicht auf Disk!). Dieses Feld arbeitet nur bei gleichzeitig eingeschaltetem Find-Feld.
Boot-Feld: vor dem Einladen des Bootblocks einer Bootdiskette wird eine Kopie der Exec-Base und anderen wichtigen Registern erstellt. Nach dem Ausführen des Bootblocks wird dann auf verdächtige Veränderungen getestet und gegebenenfalls ins Modul gesprungen, damit der Anwender in einem erweiterten Virenmenü verschiedene Möglichkeiten hat. Diese Funktion macht mit manchen autobootenden Festplatten Probleme. Falls dennoch mit einer solchen Festplatte gearbeitet werden soll, muß diese Funktion deaktiviert werden (nur Boot-Feld).

Erkennt das Modul bei einem Bootvorgang einen Virus, so gelangen Sie automatisch in ein Menue, in dem Ihnen folgende Punkte zur Auswahl stehen:

F1 - Continue booting: setzt den Bootvorgang fort, vergewissern Sie sich aber vorher, daß sich dadurch kein Virus in Ihren Amiga einschleicht (s.u.)
F2 - Display bootblock: zeigt den Bootblock als ASCII-Text an. Viren verraten sich dabei oft durch eindeutige Texte im Bootblock.
F3 - Install normal bootblock: Installiert auf der eingelegten Diskette einen herkömmlichen Bootblock - der darauf befindliche Virus wird gelöscht.
F4 - Install anti-virus-boot: Installiert auf der eingelegten Diskette einen Bootblock, der neben der gewöhnlichen Aufgabe eines Bootblocks, auch den Speicher nach Viren durchsucht und gefundene Viren ausgibt. Dies ist in jedem Fall einem normalen Bootblock vorzuziehen!
F5 - Restore execbase and continue booting: trägt wieder die herkömmlichen Werte in die ExecBase-Struktur des Amiga ein. Dadurch werden die meisten Viren im Speicher gelöscht.
F6 - Show changed vectors: gibt die ExecBase-Vektoren aus, die durch den Bootblock verändert wurden.
F10- Exit to monitor: Der normale Modul-Editor wird aufgerufen, so daß Ihnen alle Modul-Befehle zur Verfügung stehen. Nach dem X-Befehl wird der Bootvorgang normal und ohne weitere Überprüfung fortgesetzt.

4.2.6 Mitte Unten: SAFE DISK

Resident-Feld: Falls aktiviert wird das SafeDisk-Tool bei jedem RESET automatisch installiert, wobei sämtliche Funktionen des Tools aktiviert werden. Die NoClick-Funktion kann aber auf Wunsch abgeschaltet werden.
NoClick-Feld: Schaltet die NoClick Funktion des SafeDisk-

Tools bei automatischer Einbindung beim RESET an oder aus.

4.2.7 Rechts Oben: Setmap D

Falls aktiviert wird bei jedem RESET automatisch die deutsche Tastatur-Belegung installiert, so daß ein Aufrufen des Setmap-Befehls der WorkBench entfällt.
Diese Funktion arbeitet nur bei gleichzeitig eingeschaltetem Bootvirustest (s.VIRUSTEST Boot-Feld)

4.2.8 Unten Mitte: LOAD und SAVE

dienen zum Einladen (LOAD-Feld) und Abspeichern (SAVE-Feld) der Preferences-Einstellungen beider Preferences-Seiten und der eingestellten (Modul-Editor) Tastaturbelegung, sowie des Savequick Bereiches (s. SQMEM-Befehl).

4.2.9 Unten Rechts: OK und NEXT PAGE

Beim Anklicken von OK wird die Preferences-Seite verlassen und die Einstellungen übernommen. (Gleiche Funktion wie ESC-Taste)
Beim Anklicken von NEXT PAGE wird auf die erste Preferences-Seite umgeschaltet (Gleiche Funktion wie SPACE-Taste)

4.2.10 Rechts Mitte: Burstnibbler

Beim Anklicken von FASTSTART wird, falls aktiviert, bei jedem Reset überprüft ob die linke Maustaste gedrükt worden ist, in diesem Fall wird sofort das Burstnibbler Kopierprogramm gestartet.

5. Beschreibung der Modul-Editor-Befehle

Zum besseren Verständnis der folgenden Befehlsbeschreibung erläutern wir nun einige wichtige und oft gebrauchte Fachbegriffe:

Syntax
  Die Syntax eines Befehles ist die genaue Beschreibung des Befehles und seiner möglichen Parameter.
Parameter
  Ein Parameter ist ein Zahlenwert oder ein String, welcher an einen Befehl angehängt wird.
Adresse
  Eine Adresse bedeutet einen Zeiger (Zahl) auf den Physikalischen Speicher (RAM/ROM) des Amigas. Um Fehlbedienungen zu erschweren werden nur folgende Adressbereiche als gültig anerkannt:
    $000000 - $100000 : Chip-RAM (bis max. 1 MB)
    $200000 - $400000 : externe Speichererweiterungen

```
    $600000 - $A00000 :   -"-          -"-
    $C00000 - $C80000 : interne 512 KB Speichererweiterung
    $C00000 - $D00000 : interne 1 MB Speichererweiterung
    $C00000 - $DC0000 : interne 1.8 MB Speichererweiterung
    $DC0000           : Echtzeituhr
    $FC0000 - $FFFFFF : Kickstart-ROM
```

Zahlenwert

Ein Wert ist eine Zahl zwischen $0 und $ffffffff, d.h. eine ganze Zahl (mit/ohne) Vorzeichen.

String

Ein String ist eine Zeichenfolge eingeschlossen von Anführungszeichen (können bei AmigaDOS - Dateinamen weggelassen werden). Wenn ein String verlangt ist, kann dieser aber auch mit Daten gemischt werden. z.B.:

1. "DIES IST EIN STRING"
2. "DIES" $20 "IST" $20 "AUCH EIN STRING"
3. $20 $21 $22 $45 $56

Alle drei Beispiele stellen Strings dar, allerdings mit unterschiedlicher Länge und Inhalt.

PC

Bei dieser Bezeichnung handelt es sich um die Abkürzung für (P)rogram-(C)ounter. Der PC enthält die Adresse des Maschinenbefehls, den der Prozessor (beim Amiga ein M68000) als nächstes ausführen würde.

Task

Ein Task ist nichts anderes, als ein Programm. Da der Amiga das Multitasking beherrscht, schaltet er schnell zwischen internen Tasks (-Programmen) um. Davon merkt der Anwender allerdings nichts und glaubt, daß alle Programme gleichzeitig ablaufen.

CPU

Der Begriff CPU ist die Abkürzung für (C)entral-(P)rocessing-(U)nit. Dahinter verbirgt sich nichts anderes als der Mikroprozessor des Computers (beim Amiga 500&2000 ein M68000)

ACHTUNG:

- Wenn bei den nachfolgenden Befehlsbeschreibungen in der Syntax ein Parameter in runden Klammern steht, so kann er weggelassen werden!
- Wenn in der Syntax das Oder-Zeichen "|" steht, gibt es mehrere Auswahlmöglichkeiten, aber nur eine von ihnen darf angegeben werden.
- Wenn in der Syntax ein Parameter nicht in runden Klammern steht muß er angegeben werden.
- Bei Befehlen, die auf AmigaDOS-Files zugreifen, können die Filenamen und -Pfade weggelassen werden: es erscheint dann eine FileRequester-Box
- Bei sämtlichen Zahleneingaben (außer Preferences u.ä.) können anstatt Zahlen auch mathematische Ausdrücke verwendet werden (siehe Zahleneingaben)

5.1 Freezer und Ripper - Befehle

5.1.1 SA - Befehl Freezen auf Diskette

Syntax: a) SA (path)name(, crunchrate)
        b) SR (path)name(, crunchrate)

zu a)
speichert das aktuelle Programm unter dem Namen name im Verzeichnis path oder dem aktuellen Verzeichnis (s. CD - Befehl) ab. Es werden dabei automatisch alle wichtigen Daten auf Ihre AmigaDOS-Diskette abgespeichert. Vor dem Speichern der Daten werden diese vom Modul komprimiert (erkennbar an der Farbveränderung von schwarz -> rot -> blau -> grün -> schwarz). Die Effizienz des Packers kann mit der eventuell angegebenen crunchrate beeinflusst werden. Dabei wird um so besser gepackt (dauert aber länger!), je größer die angegebene Zahl (immer zwischen $10 und $7fff) ist. Standardmäßig eingestellt ist $20.
Falls das Programm mehr als 1024kB belegt (einstellbar mit Preferences), erfolgt eine Sicherheitsabfrage, ob ein so großes Programm wirklich gefreezt werden soll, da dann das abgefreezte Programm u.U. mehrere Disketten lang sein kann. Falls Sie also mehr als 1024kB abfreezen wollen, sollten Sie genügend formatierte leere Disketten bereithalten. (Faustregel: 1 Diskette pro zu speicherndem Mb Speicher)
Wir empfehlen Ihnen aber zuerst zu versuchen, ob das abzuspeichernde Programm nicht auch ohne Speichererweiterungen lauffähig ist. Dazu schalten Sie Ihre Speichererweiterungen wie im Kap. Preferences beschrieben aus und booten Sie neu.

zu b)
gleiche Funktion, wie der SA - Befehl, jedoch wird nach Beendigung des Abspeichervorgangs automatisch das Modul verlassen und das unterbrochene Programm fortgesetzt. (Funktion wie SA - Befehl mit anschließendem X - Befehl)

```
Bsp.: SA "0:game"       -speichert auf Laufwerk df(0) ab
      SA 0:game         -ebenso
      SA spiel,30       -effektiverer Packvorgang
      SR 0:game         -freezen und fortsetzen
```

5.1.2 LA - Befehl Einladen eines gefreezten Files

Syntax: a) LA (path)name
        b) LR (path)name

zu a)
lädt das gefreezte File name aus dem Verzeichnis path (oder dem aktuellen) von Diskette ein. Das so geladene Programm kann dann mit dem X-Befehl gestartet werden.
Falls das zu ladende Programm über mehrere Disketten verteilt ist, werden Sie jeweils aufgefordert, die entsprechende Diskette einzulegen.

zu b)
wie LA-Befehl, jedoch wird nach dem Einladen und Entpacken des Programms dieses sofort gestartet. (Funktion wie LA-Befehl mit anschließendem X-Befehl)

Bsp.: LA 2:ordner/game - lädt vom (df)2: aus dem Verz. ordner das gefreezte File "game"
LR 2:ordner/game - wie oben, setzt aber das eingeladene Programm fort

5.1.3 SLOADER - Befehl Abspeichern des Ladeprogramms

Syntax: SLOADER

speichert ein vom AmigaDOS CLI aus ausführbares Programm ab (ins aktuelle Verzeichnis s. CD - Befehl), mit dem Sie vom Modul gefreezte und abgespeicherte Programme auch ohne Modul wieder einladen können. Insbesondere können Sie dieses Ladeprogramm, das unter dem Namen ALOAD abgespeichert wird, z.B. in das C: Verzeichnis Ihrer Festplatte kopieren (erst auf Diskette und dann vom CLI aus mit dem COPY Befehl auf Ihre Festplatte).
Um ein Programm dann ohne Modul zu laden, müssen Sie von einem AmigaDOS-CLI aus 'ALOAD name' eingeben, wobei Sie für name den Namen (mit Pfad) des abgespeicherten Programms eingeben.
Sollte das zu ladende File auf mehrere Disketten verteilt abgespeichert worden sein, werden Sie vom Programm aufgefordert die entsprechenden Disketten einzulegen. Wenn Sie die jeweils geforderte Diskette eingelegt haben, drücken Sie die linke Maustaste. Mit der rechten Maustaste können Sie bei dieser Gelegenheit den Ladevorgang abbrechen.

Bsp.: SLOADER - Speichert den Autoloader ins aktuelle Verzeichnis

Um ein abgefreeztes Programm zu laden und starten (auch ohne Modul) vom CLI aus eingeben: ALOAD name

5.1.4 SQ - Befehl Abspeichern auf die RAM-Disk

Syntax: a) SQ
b) SQR

zu a)
speichert das gerade ausgeführte Programm nicht auf Diskette, sondern in den (unbenutzten) Speicher Ihres Amigas, wobei ein eventuell zuvor (mit SQ-Befehl) abgespeichertes Programm überschrieben wird. Dies geht wesentlich schneller, als das Abspeichern auf Diskette (SA-Befehl), hat jedoch den

Nachteil, daß das abgespeicherte Programm nach Abschalten des Computers verloren ist.

zu b)
wie SQ-Befehl, startet jedoch nach dem Abspeichern das Programm wieder automatisch (Funktion wie SQ-Befehl mit anschließendem X-Befehl)

**ACHTUNG: Dieser Befehl funktioniert nur, wenn Sie einen Amiga besitzen, der über mindestens 1 Mb RAM verfügt (z.B. A2000 oder A500+interne Speichererweiterung) und dem Amiga beim booten jedoch nur 512kB oder 1Mb zur Verfügung standen. Dies erreichen Sie mit Hilfe der Speicherabschaltung der Preferences - Seite des Moduls (F3-Taste)!
Dabei muß der freie Speicher aus 512kB bzw. 1Mb bestehen, die "am Stück" liegen. (s. auch SQMEM-Befehl)**

Bsp.: SQ oder SQR

5.1.5 SQMEM - Befehl Speicher für SQ-Befehl zuweisen

Syntax: SQMEM (adresse)

Ein unbenutzter Speicherbereich wird für den SQ-Befehl reserviert.

Bsp.: SQMEM 200000

In diesem Beispiel haben Sie eine externe 2Mb Speichererweiterung, welche mit Hife des Preferences Menue abgeschaltet wurde (siehe Preferences). Wenn Sie nun den SQ-Befehl eingeben wird das aktuelle Programm nach der Adresse $200000 abgespeichert.

5.1.6 LQ - Befehl Einladen vom RAM

Syntax: a) LQ
b) LQR

zu a)
lädt ein Programm vom unbenutzten RAM Ihres Amigas wieder ein, welches dann mit dem X-Befehl gestartet werden kann. Zuvor muß jedoch ein solches Programm mit Hilfe des SQ oder SQR-Befehls abgespeichert worden sein.

zu b)
wie LQ - Befehl, jedoch wird nach dem Einladen des abgespeicherten Programms dieses gestartet. (Funktion wie LQ-Befehl mit anschließendem X-Befehl)

Bsp.: LQ oder LQR

5.1.7 EXQ - Befehl Savequick File <--> akt. Programm

Syntax: a) EXQ
        b) EXQR

zu a)
tauscht das aktuelle Programm mit dem in der Ram-Disk gespeicherten Programm (siehe SQ-Befehl) aus.

zu b)
siehe EXQ-Befehl, jedoch wird nach dem Austauschen das neue Programm sofort gestartet.

Bsp.: EXQ oder EXQR

5.1.8 TRACKER - Befehl Sucht nach Musikstücken

Syntax: TRACKER (start)
  Kurz: SRIP (start)

durchsucht das ChipRAM (evtl. ab der mit start angegebenen Adresse) des Amigas nach Musickstücken im Soundtracker-Format. Dabei wird nach drei verschiedenen Algorithmen gesucht. Der jeweilige Stand der Suche wird auf dem Bildschirm ausgegeben.
Wird der Befehl fündig, so steht Ihnen ein weiteres Menue zur Verfügung:

F1 - spielt das gefundene Stück
F2 - stoppt das Abspielen wieder
F3 - gibt Informationen über die benutzten Instrumente und Samples, wie z.B. Name, Länge usw.
F4 - dient zum Abspeichern des Musikstückes auf Diskette. Nach Drücken dieser Taste müssen Sie den Namen des Stückes eingeben (evtl. mit Pfad), unter dem es auf Diskette gespeichert werden soll (+ RETURN). Falls Sie keinen Namen angeben, wird der FileRequster aufgerufen. So abgespeicherte Musikstücke können dann mit den meisten üblichen Soundtrackern wieder eingeladen und weiterverwendet werden.
F6 - gibt während dem Abspielen des Stücks die aktuell gespielten Musikdaten aus
F7 - setzt die Suche nach weiteren Musikstücken fort.
F8 - verändert das Musikstück so, daß es auch mit Soundtrackern wieder eingeladen werden kann, die nur 16 Instrumente verarbeiten können.
F9 - korrigiert eventuell falsch im Speicher stehende Pattern-Längen, so daß das Musikstück vom Modul aus richtig abgespielt und abgespeichert wird.

F10- beendet den Befehl und Sie befinden sich wieder im Modul - Editor

Am Besten funktioniert der TRACKER-Befehl mit Musikstücken aus INTROS und DEMOS, da diese oft standardisierte Musikdaten benutzen, die vom Modul gefunden werden können, aber auch bei einigen Spielen kann er die Musik finden.

Bsp.: TRACKER

5.1.9 SCAN - Befehl Suchen nach digitalisierten Samples

Syntax: SCAN

stellt das ChipRAM des Amigas als Klangkurve dar und bietet mittels eines Menues weitere Möglichkeiten:

Maustaste - verschieben der Start- (linke Maustaste) bzw. Endadresse (rechte Maustaste) des abzuspielenden Samples.
Cursor rechts/links - verschieben der Startaddresse (bzw. bei gedrückter SHIFT-Taste der Endaddresse) des abzuspielenden Samples
F1 - spielt den Sample vom eingestellten Start bis zum Ende mit der eingestellten Frequenz (Sampleperiod) ab.
F2 - zoomt den eingestellten Bereich graphisch, so daß dieser detaillierter betrachtet werden kann
F3 - setzt die Parameter Start, Ende und Sampleperiod wieder auf die anfänglichen Werte zurück.
F4 - vergrößert den dargestellten (graphischen) Bereich um mehr Übersicht zu erlangen (aus-zoomen)
F5 - speichert den eingestellten Sample als IFF-8svx File ab, das von Musikprogrammen wieder eingeladen werden kann, die dieses standard Format verstehen.
1-4 - mit den Zahlentasten 1, 2, 3 und 4 werden die Parameter Start, Ende und Sampleperiod auf die Werte des aktuell gespielten Samples des Kanals 1-4 gestellt. Damit können Sie diese Sounds besonders schnell finden (oft durch aus-zoomen nach drücken der Zahlentasten)

Bsp.: SCAN

5.1.10 SP - Befehl Abspeichern von Bildern im IFF-Format

Syntax: SP (path)name(, picnr (höhe))

speichert das aktuelle Bild unter dem Namen name in das Verzeichnis path (oder das aktuelle) ab. Bei manchen Bilder, oft auch bei Spielen, kommt es vor, daß das Bildschirmbild sich aus mehreren Bildern zusammensetzt, ähnlich sich über-

lappenden Workbenchscreens. Diese können nur einzeln gespeichert werden. Welches Bild man nun speichern möchte, gibt man mit picnr an. läßt man die picnr weg, wird das erste Bild abgespeichert.
Falls die Bildhöhe von Modul nicht richtig erkannt worden sein sollte, oder mit Hilfe des P-Befehls eine eigene gewählt wurde, kann diese mit Bildhöhe angegeben werden. Das abzuspeichernde Bild sollte vorher mit dem P-Befehl begutachtet werden!
Das Bild, welches der P-Befehl darstellt, wird 1:1 auf die Diskette abgespeichert
Bilder, die im Dual-Playfield Modus dargestellt werden, einem Amiga-eigenen Darstellungsmodus, bei dem zwei unterschiedliche Bildschirme transparent übereinandergelegt werden können werden in die zwei einzelnen übereinandergelegten Bilder unterteilt und getrennt abgespeichert. Dies wird im Datei-Namen durch ein angehängtes ".1" oder ".2" (für den ersten und zweiten Bildschirm) kenntlich gemacht.

Bsp.: SP "1:bild",1 !200 - Speichert unter dem Name "bild" im Laufwerk df(1) das erste Bild mit der Bildhöhe von 200 Pixeln ab.

5.1.11 P - Befehl Suchen/Darstellen von Bildern

Syntax:nr)

stellt das aktuelle Bild dar. Bei manchen Bildern, oft auch bei Spielen, kommt es vor, daß das Bildschirmbild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Diese können nur einzeln angezeigt werden. Welches Bild man nun sehen möchte, gibt man mit picnr an. Während der Bilddarstellung kann nun mit der rechten und linken Maustaste der untere Rand, eingestellt werden. Bei Verlassen der Bilddarstellung mit der ESC-Taste, wird dann die Höhe des so gewählten Bildes ausgegeben. Dieser Wert sollte beim Abspeichern des Bildes mitangegeben werden. Durch die Angabe der Bildhöhe beim Speichern haben Sie Gewissheit, daß nur das Bild abgespeichert wird und keine überschüssigen Daten. Außerdem stehen Ihnen während der Bilddarstellung noch folgende Befehle auf Tastendruck zur Verfügung:

a - autoplane:	setzt Planepointer mit Offset
b - brightness plus:	hellt das Bild auf
B - brightness minus:	dunkelt das Bild ab
c - colorreg plus:	Farbregister plus 1
d - dual playfield on:	Dualplayfieldmodus einschalten
D - dual playfield off:	Dualplayfieldmodus ausschalten
e - right border plus:	rechter Rand plus 1
E - right border minus:	rechter Rand minus 1
f - fast plane up:	ein Bild nach oben blättern
F - fast plane down:	ein Bild nach unten blättern
g - interlace on:	schaltet Interlacemodus ein
G - interlace off:	schaltet Interlacemodus aus

h - hold and modify on:	HAM - Modus ein
H - hold and modify off:	HAM - Modus aus
i - invert all colors:	invertiert alle Farbregisterwerte
l - lores mode on:	niedrige Auflösung (40 Zeichen) ein
L - lores mode off:	hohe Auflösung (80 Zeichen) ein
m - modulo 1+2 plus:	erhöht beide Modulowerte um 2
n - modulo 1+2 minus:	erniedrigt beide Modulowerte um 2
o - modulo 1 minus:	erniedrigt Modulowert 1 um 2
O - modulo 2 minus:	erniedrigt Modulowert 2 um 2
p - modulo 1 plus:	erhöht Modulowert 1 um 2
P - modulo 2 plus:	erhöht Moduloewrt 2 um 2
q - clear modulo 1+2:	setzt beide Modulowerte auf null
r - rotate planepointer:	rotiert die Planepointer
s - left border minus:	linker Rand minus 1
S - left border plus:	linker Rand plus 1
w - white helpscreen:	weisse Parameterleiste
W - black helpscreen:	schwarze Parameterleiste
x - colorreg minus:	Farbregister minus 1
y - switch diw and ddf mode:	Wähle Scrollmodus
+ - add plane:	erhöht die Planezahl
- - sub plane:	erniedrigt die Planezahl
= - set planepointer to one:	setzt alle Planepointer auf den Wert des ersten.
0 - unlock all planes:	gibt alle Planes frei
SHIFT 0 - lock all planes:	wählt alle Planes fest (nur auf dem 10er-Block!)
1-6 unlock plane:	wählt jeweilige Plane fest
SHIFT 1-6 unlock plane:	gibt jeweilige Plane frei (nur auf dem 10er-Block!)
7 - red color plus:	
8 - blue color plus:	
9 - green color plus:	erhöht Farbwert des eingestellten Farbregisters
SHIFT 7,8,9:	erniedrigt Farbwert des eingestellten Farbregisters (nur auf dem 10er-Block!)
CURSORTASTEN:	rotieren/scrollen des Bildes in die jeweilige Richtung
SHIFT UP/DOWN:	scrollen des Bildes schneller hoch und runter
DEL - hide helpscreen:	schaltet die Parameterleiste ab
HELP - show helpscreen:	schaltet die Parameterleiste ein
F1 - default colors:	setzt Standardfarben
F2 - random colors:	setzt Zufallsfarben
F10 - set current picture:	setzt Bildparameter ins laufende Programm, diese können jedoch im Einzelfall wieder von der aktiven Copper-Liste überschrieben werden, so daß die Veränderungen nicht von Dauer sind.

Mit der Maus kann die eventuell eingeschaltete Parameterleiste beliebig auf dem Bildschirm postiert werden.
Um das jeweils auf dem Schirm angezeigte Bild abzuspeichern verlassen Sie den P-Befehl mit der ESC-Taste und benutzen Sie den SPM-Befehl (siehe unten)

5.1.12 SPM - Befehl Verändertes Bild abspeichern

Syntax: SPM (path)name

falls Sie mit Hilfe des P-Befehls das aktuelle Bild verändert haben, können Sie das VERÄNDERTE Bild mit Hilfe des SPM-Befehls unter dem Namen name abspeichern

Bsp.: SPM ordner/test - Speichert das Bild aus dem Mempeeker (P - Befehl) unter dem Namen "test" im Directory "ordner" auf das aktuelle Laufwerk ab.

5.2 Trainer - Befehle

5.2.1 T - Befehl Trainermaker

Syntax: a) TS wert
b) T (wert)
c) TX

zu a)
startet den TrainModus und beginnt die Suche nach dem angegebenen Wert. Der angegebenen Wert stellt den Zählerstand dar, nach dem Sie suchen.
Mit dem TS/T - Befehl können Sie die Adresse des von Ihnen gesuchten Zählers erhalten, mit dieser Information ist es Ihnen dann möglich z.B. mit dem M-Befehl sich nach Bedarf mit Leben zu versorgen.
Wenn Sie einmal den Trainer gestartet haben sind Sie im "Trainmode".

zu b)
setzt den Trainvorgang fort mit dem angegebenen Wert. Adressen, die verdächtig sind, der gesuchte Zähler zu sein, werden ausgegeben. Wenn der Trainer nichts gefunden hatte wird die Meldung "TRAINERMAKER WAS NOT SUCCESSFULL!" ausgegeben.
In diesem Fall starten Sie bitte einen neuen Trainvorgang oder beenden Sie den Trainer mit dem TX-Befehl
Läßt man den Parameter Wert weg, so werden die zuletzt gefundenen Adressen wieder ausgegeben.

zu c)
Beendet den Trainermodus

Bsp.: TS !3
FIRST TRAINPASS!
CHANGE COUNTVALUE NEXT TIME!
SEARCHED UP TO: 005444
TRAINMODE AKTIVE!
READY.
X

T !2
SEARCHED UP TO: 080000
00014428
TRAINMODE AKTIVE
READY.

In diesem Beispiel hatten Sie am Anfang 3 (Leben), danach verließen Sie das Modul (X-Befehl) und verloren (absichtlich) 1 (Leben) und hatten nun nur noch 2 (Leben). Der Modul-Trainermaker hat nun als Addresse des Zählers 0014428 ausgegeben. Verändern Sie nun die bei Ihnen natürlich andere Adresse mit dem M-Befehl (im Bsp.: M 14428 + RETURN) und tragen Sie den neuen Wert (z.B. 09) ein. Nun verlassen Sie das Modul abermals und erfreuen sich Ihrer 9 (Leben).
Oder aber fahren Sie fort wie folgt:

TX
TRAINMODE INAKTIVE!
TFD 14428
SUB FOUND AT 001122
SUBS ELIMINATED!

Um sich beliebig viele (Leben) zu verschaffen.

Zur genauen Handhabung des Trainers siehe auch Anhang A "Handhabung des Trainers"

5.2.2 TD - Befehl Deep Trainer (findet fast alles)

Syntax: a) TDS
b) TDC
c) TDX
d) TD
e) TDI
f) TDD start end

Den neuen Deep-Trainer sollte man in den Fällen benutzen, wo man mit dem alten Trainer kein Resultat erhält oder den alten Trainer nur schlecht anwenden kann. Der neue Deep-Trainer benötigt jedoch einen freien, vom Programm unbelegten Speicherbereich. Der Hauptvorteil des Deep-Trainers ist, daß er keinerlei Zahlenwerte benötigt und daher auch bei relativ unbestimmbaren Werten wie z.B. bei Energiebalken hilft.

Siehe auch Anhang C

zu a)
Start und fortsetzen des Deep-Train Vorgangs. Dabei sollte man auf einen Wert achten der reproduzierbar ist, wie z.B. einen vollen Energiebalken. Es werden danach mögliche Adressen ausgegeben. Bei zuvielen oder keiner Adresse(n) sollte man mit a) oder b) fortsetzen.

zu b)

Fortsetzen des Deep-Train Vorgangs, wobei die gesuchte Größe einen unterschiedlichen Wert zu dem Startwert und dem letzten TDC Wert haben muß.

zu c)
Beendet den aktuellen Deep-Trainer. Sie können nun einen neuen Deep-Trainer mit TDS starten.

zu d)
zeigt alle gerade im Moment untersuchten Adressen an.

zu e)
zeigt die verdächtigsten Adressen an, die bis zu diesem Zeitpunkt untersucht wurden.

zu f)
mit diesem Befehl können Sie einige der gerade untersuchten Adressen von start bis ende löschen, d.h. von einer weiteren Untersuchung ausschließen.

5.2.3 TF - Befehl Absoluter Trainer (beliebig viele Leben)

Syntax: a) TF adresse (start (ende))
        b) TFD adresse (start (ende))

zu a)
zeigt die Adressen des Programmes an, wo wahrscheinlich mit dem Befehl "SUBQ.X #X,adresse" oder "SUBI.X #X,adresse" der Inhalt der angegebenen Adresse verringert wird (findet auch indirekte Adressierungsarten).
Dabei wird der gesamte, oder angegebene Speicherbereich durchsucht.

zu b)
wie a) aber zusätzlich werden diese Maschinenbefehle aus dem Programm entfernt -> Das laufende Programm zieht von dem in der Adresse enthaltenen Zähler nichts mehr ab!
Dabei wird der gesamte, oder angegebene Speicherbereich durchsucht.

Bsp.: siehe TS-Befehl

5.2.4 PC - Befehl Messe Energiebalken aus

Syntax: siehe P-Befehl

stellt das aktuelle Bild dar. Bei manchen Bildern, oft auch bei Spielen, kommt es vor, daß das Bildschirmbild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Diese können nur einzeln angezeigt werden. Welches Bild man nun sehen möchte, gibt man mit picnr an. Läßt man picnr, wird das erste Bild dargestellt. Während der

Bilddarstellung können Sie mit Hilfe der Maus einen Bildausschnitt wählen, dessen Höhe und Breite in Bildschrimpunkten nach drücken der ESC-Taste ausgegeben wird. Damit ist es ein leichtes die Länge von Energiebalken auszumessen und diesen Wert im Zusammenhang mit dem Trainer weiterzuverwenden (Zähler).
Drückt man die linke Maustaste, kann man die linke obere Ecke des Auschnittes verschieben. Drückt man die rechte Maustaste, kann die rechte untere Ecke verschoben werden.
Um für den Trainer verwertbare Resultate zu erzielen, sollte man darauf achten, den Bildauschnitt "pixelgenau" zu setzen!

```
Bsp.: P 1
      PICTUREHIGHT: !256
      READY

      PC 1
      WIDTH : X = !123
      HEIGTH: Y = !010
      READY
```

5.3 Anti - VIRUS - Befehle

Nachfolgend folgen zwei Befehle zur Bekämpfung von Viren/resetfesten Programmen im Speicher des Computers. Dies ist aber nur der eine Teil eines wirksamen Virusschutzes. Der wichtigere Teil ist es, Viren auf infizierten Disketten aufzuspüren und diese (die Viren) zu löschen. Das Aufspüren von Viren nimmt Ihnen von nun an das Amiga Action Replay Mk III ab: Sobald Sie Ihren Computer einschalten überwacht das Modul (sofern diese Option nicht mit Preferences ausgeschaltet wurde) sämtliche Disketten, von denen Sie booten und meldet aktivitäten, die auf Viren schließen lassen.
Die nachfolgenden Befehle ergänzen lediglich den Virusschutz und werden zum Teil auch automatisch ausgeführt.

5.3.1 VIRUS - Befehl Viren im Computer aufspüren

Syntax: VIRUS

durchsucht den Speicher nach Viren/resetfesten Programmen und zeigt diese an.
Dieser Befehl wird vom Modul automatisch beim Aufruf mittels des Modul-Tasters ausgeführt, falls die VIRUSTEST-Find-Option im Preferences Menue angewählt wurde (beim Einschalten automatisch aktiviert).

Bsp.: VIRUS

5.3.2 KILLVIRUS Viren im Computer vernichten

Syntax: KILLVIRUS
  Kurz: KVIR

sucht und entfernt Viren/resetfeste Programme aus dem Speicher des Computers. Diese Maßnahme entfernt keine Viren auf Disketten. Um Bootblock-Viren von infizierten Disketten zu entfernen siehe INSTALL-Befehl.
Dieser Befehl wird automatisch beim Aufruf mittels des Modul-Tasters ausgeführt, falls die VIRUSTEST-Kill-Option im Preferences Menue angewählt wurde.

Bsp.: KILLVIRUS

5.4 Disketten und Diskettenkodier - Befehle

Mit dem Amiga Action Replay Mk III können Sie jetzt direkt mit AmigaDOS-kompatiblen Disketten arbeiten. Sämtliche Diskettenoperationen liefern direkt Dateien im AmigaDOS-Format, manche auch im standartisierten IFF Format (Bilder,Sounds).

WICHTIG: Falls Sie an AmigaDOS-Disketten vom Modul-Editor aus irgendwelche Veränderungen vornehmen, z.B. eine Diskette formatieren, ein File löschen u.ä., so erkennt das Amiga Betriebssystem nach dem Verlassen des Moduls (X-Befehl) diese Veränderungen erst, wenn Sie die veränderte Diskette aus dem Laufwerk entfernt und wieder eingelegt haben!

Neben dieser Möglichkeit können Sie jetzt aber auch Ihre Disketten mit Codewörtern vor fremdem Zugriff schützen (auch Sicherheitskopien u.ä.). Dazu bietet Ihnen das Amiga Action Replay Mk III zwei verschiedene Möglichkeiten:

1.) Bei Programmen, die direkt vom Bootblock aus gestartet werden und bei denen der übrige Disketteninhalt in einem Fremdformat gehalten wurde (Not A DOS-Disk) und bei normalen DOS-Disketten, besteht die Möglichkeit den Bootblock durch ein Passwort zu schützen. Benützen Sie dazu die Befehle BOOTCODE und BOOTPROT

2.) Bei Programmen, die keine eigenen Disketten-Laderoutine benutzen, sonder von AmigaDOS geladen werden, kann die gesamte Diskette kodiert werden. Dazu muß die unkodierte Diskette in einem speziellen Verfahren umkopiert werden. Benutzen Sie dazu die Befehle CODE, CODECOPY.

WICHTIG: Arbeiten Sie beim Kodieren IMMER mit Sicherheitskopien!!!!!

5.4.1 BOOTCODE - Befehl Bootblockkodierer aktivieren

Syntax: BOOTCODE (codenumber)
  Kurz: BCODE (codenumber)

stellt die Kodezahl (0 - $ffffffff) codenumber für den Bootblockkodiererr ein, oder gibt die aktuelle Kodezahl aus, falls keine angegeben wurde. Ist die Kodezahl ungleich null, ist der Bootblockkodierer aktiviert, und Sie können jetzt auch von Kodierten Disketten booten. (Unkodierte Disketten können Sie auch jetzt noch normal booten)
Um Ihre Disketten zu schützen, müssen Sie diese aber auch zuvor mit dem BOOTPROT-Befehl kodiert haben (s. unten)!

Bsp.: BOOTCODE 873233

aktiviert den Bootkodierer und setzt das Kodewort auf die Zahl $873233 -> nur noch unkodierte und mit dieser Zahl kodierte Disketten können gebootet werden.

    BOOTCODE 0

schaltet den Bootcoder wieder aus -> nur noch unkodierte Disketten können gebootet werden.
(siehe auch Preferences BootBlockCoder)

5.4.2 BOOTPROT-Befehl BootBlock einer Diskette kodieren

Syntax: BOOTPROT (codenumber)
  Kurz: BPROT (codenumber)

kodiert den BootBlock der Diskette im aktuellen Laufwerk mit dem Kodewort codenumber (Zahl zw. 0 und $ffffffff). Von solcherart behandelten Disketten können Sie nur noch booten, wenn Sie mit Hilfe des Moduls das BootBlock - Kodewort auf diese spezielle Zahl einstellen (mit BOOTCODE-Befehl). Falls Sie die Kodezahl codenumber weglassen wird die momentan eingestellte (BOOTCODE-Befehl) verwendet.
Um Disketten wieder zu decodieren, wenden Sie den BOOTPROT-Befehl mit Ihrem speziellen Kodewort, mit dem Sie sie kodiert haben einfach nochmals auf die Diskette an.

Bsp.: CD 0: - wählt Laufwerk df(0) als aktuelles Laufwerk
    BOOTPROT 12348765

Diskette im Laufwerk df0: ist kodiert und kann nicht mehr gebootet werden.

    BOOTCODE 12348765

Modul wurde auf richtiges Kodewort eingestellt. Der Computer kann jetzt die immer noch kodierte Diskette wieder booten. Ohne Modul ist dies nicht mehr möglich

    BOOTPROT 13248765

Diskette ist wieder dekodiert und kann normal geladen werden.

5.4.3 CODE - Befehl Diskettenlaufwerke kodieren

Syntax: CODE (drive (codenumber))

kodiert ein bestimmtes Laufwerk (drive) mit der Kodenummer codenumber (Zahl zw. 0 und $ffff), so daß dieses Laufwerk ab sofort nur noch mit dieser Kodenummer kodierte Disketten akzeptiert. Falls keine Kodenummer angegeben wurde, wird das Laufwerk drive wieder in den normalen (unkodierten) Zustand versetzt. Die eingestellten Kodewörter bleiben auch über einen Computer Reset hinaus erhalten, so daß Sie also auch von kodierten Disketten booten können.
(siehe auch Preferences)

ACHTUNG: Kodierte Laufwerke verarbeiten nur noch kodierte Disketten korrekt. Sämtliche Diskettenoperationen auf unkodierte Disketten von kodierten Laufwerken aus führen zu Fehlermeldungen.

Bsp.: für das kodieren und Verwenden von Disketten
      CODE 0 1234
Internes Laufwerk akzeptiert nur noch mit 1234 kodierte Disketten (auch das Modul).
Zu kodierende Disk in z.B. Laufwerk df1 einlegen
Leere Disk in Laufwerk df0 (internes Laufwerk) einlegen
      CODECOPY 1 0
Jetzt ist die Originaldiskette kodiert auf der zweiten abgespeichert worden.
Die kodierte Kopie kann nun nur noch auf Laufwerken betrieben werden bei denen mittels des CODE-Befehls die Laufwerkkodierung auf 1234 gesetzt worden ist

      CODE 0
schaltet die Kodierung des Internen Laufwerks aus -> Die Kopie ist unlesbar geworden, das Laufwerk kann kann wieder normal genutzt werden. u.s.w

5.4.4 Kopieren von AmigaDOS-Disketten

Syntax: a) DCOPY source dest
        b) BURST (drive)

zu a)
kopiert die Diskette vom Laufwerk source (0-3) auf die Zieldiskette im Laufwerk dest (0-3). Auch das kopieren mit nur einem Laufwerk ist möglich. Beachten Sie, daß der DCOPY-Befehl nur für unkodierte Disketten gedacht ist. Die kodierung von dem Quell- und/oder Ziellaufwerk wird für die Dauer

des DCOPY-Befehls ausgesetzt.

Der DCOPY Befehl arbeitet IMMER mit VERIFY, d.h. die Zieldiskette wird auf Schreibfehler untersucht und diese werden gegebenenfalls entfernt. Ebenso werden Lesefehler auf der Quelldiskette erkannt und wenn möglich korrigiert. Somit lassen sich auch widerspenstige und sogar defekte Diskette noch komfortabel verwenden oder sogar reparieren.
Nach dem Kopieren einer Diskette (mit zwei Laufwerken) können Sie das aktuelle Programm wieder fortsetzen.

zu b)
startet das Kopierprogramm Burstnibbler mit der aktuellen Speicher und Laufwerks-Belegung. Der Vorteil von Burstnibbler im Vergleich zu dem DCOPY-Befehl ist, daß man mit Burstnibbler auf mehrere Laufwerke gleichzeitig sowie Fremdformate (z.B. Atari ST,IBM) kopieren kann gleichzeitig braucht das Burstnibbler Kopierprogramm weniger Zeit um eine Diskette zu kopieren. Falls Sie aber den Burstnibbler benutzen wird das gerade aktuelle Programm gelöscht. Sie können bei dem BURST-Befehl eine Laufwerksnummer angeben, in diesem Einlaufwerks-Modus können Sie dann mit 1Mb Speicher eine Disktette auf einmal einlesen, im normalen Mehrlaufwerks-Modus brauchen Sie mind. 1,5Mb um Bei einem Kopiervrgang von z.B. DF0: -> DF0: die Quelldiskette in einem Durchgang einzulesen. Der Burstnibbler kann auch direkt nach dem Reset aktiviert werden siehe Preferences BURSTNIBBLER.

Allgemeine Bedienung des Burstnibblers:
=======================================

Zum Starten des Kopiervorgangs müssen Sie das START-Icon anklicken. Dieses Icon verwandelt sich dann in ein STOP-Icon, welches nur zum vorzeitigen Abbrechen des Kopiervorgangs betätigt werden muss

Die Auswahl des Kopiermodus erfolgt durch Betätigung des MODE-Icon, mögliche Modi sind DEEP und DOS, siehe dazu Punkt B1.

Falls Sie nur bestimmte Spuren einer Diskette kopieren wollen, so können Sie diese durch Verstellen der START und END Werte eingrenzen. Es ist auch möglich nur einzelne Seiten einer Diskette zu kopieren, indem das SIDE-Icon benutzt wird. Mögliche Einstellungen sind hier: U (Oberseite), L (Unterseite) und B (beide Seiten).

Um ein Laufwerk als Source (Quelle) zu schalten, müssen Sie das zugehörige Symbol (die Disketten) solange anklicken, bis es sich grün färbt. Nur ein Laufwerk kann jeweils als Source bestimmt werden. Das (die) Destination Laufwerk(e) (Ziel) werden durch eine braune bzw. eine violette Einfärbung dargestellt, wobei violett ein Verify (Überprüfung der geschriebenen Daten) mit sich führt.

Nur für den DEEP-Modus nötig ist das SYNC-Icon.
Durch das Anklicken desselben können Sie zwischen dem Stan-

dard-Sync (spezielle Markierung auf einer Diskette; $4489) und dem INDX-Sync (Laufwerks-Index orientiertes Kopieren) wählen.
Falls Sie Erfahrung mit verschiedenen Fremdformaten (-Aufzeichnungformaten) haben, so können Sie deren SYNC-Werte auch manuell einstellen. In aller Regel ist dies jedoch nicht erforderlich, da das Programm die SYNC-Werte von Fremdformaten kennt.
Falls Sie Das Burstnibbler Programm verlassen wollen müssen Sie einen Tastatur-Reset ausführen (Ctrl-Amiga-Amiga) und nicht das Pragramm mit dem QUIT-Icon verlassen.

B1)
Starten Des Kopievorgangs:

Stellen Sie zunächst alle gewünschten Parameter ein. Dies bedeutet hauptsächlich die Auswahl der Laufwerke als SOURCE und DESTINATION sowie den gewünschten Modus. Danach gehen Sie mit der Maus auf das START-Icon und drücken die linke Maustaste. Sollte eine Fehlermeldung auftreten (z.B. DESTINATION schreibgeschützt), so erscheint eine Fehlermeldung in der Statuszeile. Sie können dann den Fehler korrigieren und den Kopiervorgang erneut starten.
Der Burstnibbler hat zwei verschiedene Möglichkeiten um Disketten zu kopieren, den DOS-Modus und den DEEP-Modus.

B2) Der DOS-Kopiermodus:

Dieser Modus erlaubt es, normale AmigaDos-Disketten sehr schnell und effizient zu kopieren (viel schneller z.B. als DiskCopy).
Dies geschieht in der Regel mit zwei Laufwerken, und zwar wird die Diskette von dem SOURCE-Laufwerk auf die Diskette in dem DESTINATION-Laufwerk kopiert. Haben Sie mehr als zwei Laufwerke zur Verfügung, so können Sie von einer SOURCE-Diskette mehrere Kopien gleichzeitig erstellen.
Ist allerdings nur ein Laufwerk vorhanden, so muss u.U. während des Kopiervorgangs die Diskette mehrmals gewechselt werden (befolgen Sie bitte die Anweisungen in der Statuszeile). Dieses eine Laufwerk muss überdies vor dem Start als Target gekennzeichnet werden (braune oder violette Einfärbung). Das mehrmalige Wechseln der Diskette entfällt wenn mehr als 512k Hauptspeicher vorhanden ist (im Mehrlaufwerksmodus von Burstnibbler d.h. mehrere Laufwerke haben eine nicht graue Einfärbung brauchen Sie mehr als 1Mb).
Auf dem Track-Display kann der Ablauf des Kopiervorgangs verfolgt worden, wobei ein grünes Kästchen symbolisiert das alles in Ordnung ist und ein braunes Kästchen das ein Lesefehler aufgetreten ist. Sollte ein Lesefehler auftreten, wird versucht diesen zu repariren.

B3) Der DEEP-Kopiermodud

Dieser Modus erlaubt es, Fremdformate und einige Kopierschutzmechanismen zu Sicherheitszwecken zu kopieren. Der DEEP-Modus braucht etwas längere Zeit um eine Diskette zu kopieren als der DOS-Modus, da er die Datenstruktur auf den Tracks erst analysieren muss. Dadurch sind eventuell auch

mehr Diskettenwechsel nötig als beim DOS-Mode, falls nur mit einem Laufwerk kopiert werden sollte.
Es ist auch möglich die meiste Software andere Computersysteme zu kopieren.
Hier sind zu nennen: Archimedes, Atari ST, IBM und andere.
Diese Fremdformate sind kopierbar, indem das SYNC aud INDX gestellt wird. Die Anwahl der Laufwerke geschieht wie beim DOS-MOde.

Bsp.: DCOPY 0 1

kopiert Diskette vom internen Laufwerk auf das Laufwerk df1:

5.4.5 CODECOPY - Befehl Kodieren von Disketten

Syntax: CODECOPY source dest
  Kurz: CCOPY source dest

kopiert die (gegebenfalls kodierte) Diskette (AmigaDOS) im Laufwerk source (0-3) auf die Diskette im Laufwerk dest (0-3) und wird dabei mit dem Kodewort des Laufwerks dest kodiert. D.h. wenn Sie z.B. eine Diskette auf das Kodewort 1234 kodieren wollen, kodieren Sie zunächst das Ziellaufwerk (dest) mit dem Befehl CODE dest 1234 (dest=0-3) und geben dann ein: CODECOPY source dest (source,dest=0-3). Die Diskette im dest Laufwerk ist dann die kodierte Kopie des Originals im source Laufwerk. Der Befehl funktioniert nur mit zwei Laufwerken!

ACHTUNG: Sollten Sie das Codewort vergessen oder verlieren kommen Sie nicht mehr an die Daten heran. Verwahren Sie also das unkodierte Original sorgfältig.

5.4.6 COPY - Befehl File kopieren

Syntax: COPY (path)sourcename,(path)destname

kopiert das File (path)sourcename nach (path)destname.
(entspricht dem CLI-Komanndo Copy)

Bsp.: COPY 0:s/startup-sequence,1:test

In diesem Beispiel wird die Startup-Sequence von df0: nach df1:test kopiert.

5.4.7. CD - Befehl Aktuelles Verzeichnis wechseln

Syntax: CD (path)

setzt den aktuellen Modul Pfad auf das Verzeichnis path.
Dabei gelten dieselben syntaktischen Regeln, wie bei normalen AmigaDOS Pfaden. EINZIGE AUSNAHME: Anstatt df0: geben Sie 0: ein. Anstatt df1: 1: u.s.w.
Falls Sie keinen Pfad angeben wird der aktuellen Pfad ausgegeben.

Bsp.: CD 1:C

wechselt den aktuellen Pfad zum C: Verzeichnis der Diskette im Laufwerk df1:

```
CD 0:
CD
```

5.4.8 DIR - Befehl Inhaltsverzeichnis anzeigen

Syntax: a) DIR (path)
b) DIRA (path)

zu a)
gibt das Inhaltsverzeichnis des aktuellen oder angegebenen Pfades aus (entspricht dem CLI-Kommando DIR).

zu b)
Bedienung wie DIR-Befehl jedoch werden auch die Inhalte von Unterverzeichnissen ausgegeben (entspricht dem CLI-Kommando "dir (path) opt a")

Bsp.: DIR 0:

gibt Inhalt von Diskette im Laufwerk df0: aus

```
CD 1:C
DIR
```

gibt Inhalt des C - Verzeichnisses im Laufwerk df1: aus.

5.4.9 MAKEDIR - Befehl Unterverzeichnis anlegen

Syntax: MAKEDIR (path)name
Kurz: MDIR (path)name

legt im Verzeichnis path oder im aktuellen ein Unterverzeichnis mit Namen name an. (entspricht dem CLI-Kommando makedir)

5.4.10 DELETE - Befehl Datei löschen

Syntax: DELETE (path)name
Kurz: DEL (path)name

löscht die Datei name im angegebenen oder aktuellen Verzeichnis (entspricht dem gleichnamigen CLI-Kommando)

5.4.11 TYPE - Befehl Datei anzeigen (ASCII)

Syntax: TYPE (path)name

stellt die Datei name im angegebenen oder aktuellen Verzeichnis als Text auf dem Bildschirm dar.
(enspricht dem CLI-Kommando Type)

Bsp.: TYPE 0:S/STARTUP-SEQUENCE

5.4.12 RELABEL - Befehl Diskettennamen ändern

Syntax: RELABEL name Kurz: REL name

Ändert den Diskettennamen der Diskette im aktuellen Laufwerk auf den angegebenen Namen.
(entspricht dem CLI-Kommando Relabel)

Bsp.: RELABEL "Neuer Name"

5.4.13 RENAME - Befehl Datei/Verzeichnis umbenennen

Syntax: RENAME (path)alt,neu
Kurz: REN (path)alt,neu

Ändert den Namen des Files/Verzeichnisses alt im Verzeichnis path auf den String neu.
(enspricht dem gleichnamigen CLI-Kommando)

Bsp.: RENAME 0:s/startup-sequence,start

5.4.14 FORMAT - Befehl Diskette formatieren

Syntax: a) FORMAT (name)

b) FORMATV (name)
c) FORMATQ (name)

zu a)
formatiert die Diskette im aktuellen Laufwerk (CD-Befehl) im AmigaDOS-Format ohne die Formatierung zu überprüfen.

zu b)
wie a), überprüft jedoch die formatierte Diskette auf Fehler. (entspricht CLI-Kommando format)

zu c)
löscht eine bereits formatierte Diskette wieder (besonders schnell, entspricht CLI-Kommando format QUICK)

So formatierte Disketten können sowohl vom Modul als auch vom Amiga normal verwendet, aber nicht gebootet werden. Um Disketten bootfähig zu machen benutzen Sie bitte den INSTALL-Befehl des Moduls (mit Virusschutz!)

Bsp.: FORMATV LEEREDISK
FORMATV "LEEREDISK"
FORMAT

5.4.15 INSTALL - Befehl Diskette installieren

Syntax: INSTALL (bootblocknr)
Kurz: INST (bootblocknr)

installiert einen Bootblock auf der Diskette im aktuellen Laufwerk (CD-Befehl). Als Bootblöcke stehen Ihnen zwei zur Auswahl: bootblocknr = 0 -> normaler Bootblock, wie er auch vom AmigaDOS verwendet wird, bootblocknr = 1 -> Virusprotector Bootblock, der verdächtige Programme anzeigt und beim booten den Bildschirm hellgrün färbt, falls nichts verdächtiges gefunden wurde.
Da der Virusprotector mit den Kickstartversionen 1.2 und 1.3 ohne weiteres zusammenarbeitet, ist es sinnvoll stets diesen Bootblock (1) zu installieren, was einen zusätzlichen Schutz vor Viren garantiert (Bildschirm wird beim booten nicht grün: Gefahr).

Bsp.: CD 1:
INSTALL 1

installiert den Virusprotector-Bootblock auf der Diskette im Laufwerk df1:

5.4.16 DISKCHECK - Befehl Diskette nach Fehlern untersuchen

Syntax: DISKCHECK (drive)
Kurz: DCHK (drive)

untersucht das angegebene Laufwerk (oder das aktuelle) nach Fehlern in der DOS-Struktur. Die Fehler werden tabellarisch aufgelistet.

Bsp.: DISKCHECK 2

Diskette im Laufwerk df2: überprüfen

5.4.17 DISKWIPE - Befehl Diskette schnell löschen

Syntax: DISKWIPE (drive)
Kurz: DWIPE (drive)

löscht alle Daten auf einer Diskette im angegebenen Laufwerk (oder aktuellem) besonders schnell und gründlich, indem jeder Track "zerstört" wird. So behandelte Disketten können erst nach dem Formatieren mit dem FORMAT-Befehl wieder verwendet werden.

Bsp.: DISKWIPE
DISKWIPE 0
DISKWIPE 3

5.4.18 SAFEDISK - Befehl Trackdisk.device patchen

Syntax: SAFEDISK (a)(b)(s)(n)(v)(u)(q)
Kurz: SDISK (a)(b)(s)(n)(v)(u)(q)

SAFEDISK installiert im Amiga Betriebssystem ein Programm, das das Arbeiten mit Disketten sicherer und komfortabler gestaltet. Dabei kann man unter folgenden Funktionen wählen:

b - Entfernt Fehler in den Diskettenroutinen des Betriebssystems
n - Unterdrückt das lästige Laufwerksklicken, wenn keine Diskette eingelegt ist. Dies funktioniert nicht mit jedem Laufwerk. Falls ungewöhnliche Klickgeräusche zu hören sein sollten, sollte diese Funktion nicht verwendet werden!
u - Schreibt den aktuellen Track des Trackbuffers auf die Diskette und schaltet den Laufwerksmotor aus, falls längere Zeit keine Zugriffe auf diesen Track erfolgt waren.
v - Schribzugriffe auf Disketten werden einem Verify unterzogen, d.h. auf korrektes Ausführen überprüft. Falls ein Schreibfehler aufgetreten war, meldet sich ein Requester, mit dem man wählen kann, zwischen Ignorieren (sollte man nicht) und einem erneuten Schreibversuch.
s - Sollte ein Track auf einer Diskette fehlerhaft sein, versucht SAFEDISK diesen Track zu reparieren und möglichst viele Daten auf dem fehlerhaften Track zu

retten.
a - Schaltet alle oben genannten Funktionen ein.
q - Entfernt das Programm aus dem Amiga Betriebssystem.

Diese Funktion kann auch automatisch bei jedem RESET ausgeführt werden. (siehe dazu Preferences SAFEDISK)

Bsp.: SAFEDISK a ; aktiviert alle Funktionen

5.5 Diskettenmonitor - Befehle

5.5.1 RT - Befehl Tracks lesen

Syntax: RT strack (num (dest))

liest vom aktuellen Laufwerk beginnend mit strack (0-!159) soviele Tracks ein, wie mit num angegeben (einen falls nichts weiter angegeben wurde). Die Daten werden dabei nacheinander ab der Adresse dest im Speicher des Amiga abgelegt.
Wird als Zieladdresse dest nichts angegeben, so wird ein möglichst großer freier Bereich im Amiga Speicher gesucht. Dieser Bereich ist dann für spätere Diskettenzugriffe reserviert.
Der Vorteil dieses Diskettenpuffers liegt darin, daß bei Verwendung desselben keine Daten im Speicher verlorengehen, also der Computer nach dem X-Befehl nicht nach Indien reist, sondern weiterhin seinen Dienst tut, als wäre nichts passiert.
Damit dies funktioniert, wird bei Verwendung eines Diskettenpuffers jedesmal beim Verlassen des Moduls der belegte Speicher wiederhergestellt. Das merkt man daran, daß bei Eingabe von X<RETURN> eine Sicherheitsabfrage erscheint, ob man das Modul verlassen will, ohne den Diskettenpuffer zurückzusetzen, was man im Normalfall mit N<RETURN> oder auch nur mit der RETURN-Taste quittiert. Lediglich für den Fall, daß die Diskettenpufferdaten nicht gelöscht werden sollen, kann man dies mit Y<RETURN> verhindern, was aber zu Systemabstürzen führen kann aus bekannter Ursache.

IM ZUSAMMENHANG MIT DIESEM DISKETTENPUFFER wurde ein neues Zahlenformat entwickelt, das einem das lästige Umrechnen vom Tracks, Sektoren und Offsets auf die tatsächliche Adresse im Speicher erleichtert. Und zwar kann bei allen Befehlen (insbesondere den Monitorbefehlen) anstelle von Adressen auch Track, Sektor und Offset eingegeben werden:

Syntax: a) Ttrack(Ssector(Ooffset))
b) Ssector(Ooffset)

Beispiele für Diskettennamen (Rootblock Track !80 Sektor 0 Offset !433) gemäß Syntax a):

T!80S00!433 oder T50S001B1 oder T!80S001B1 u.s.w.

gemäß Syntax b):

S!8800!433 oder S37001B1 u.s.w.

Solche Ausdrücke werden vom Modul stets als Adressen interpretiert und zwar werden die Adressen folgendermaßen gewonnen:

a) Adresse = Start des Diskpuffers + track*(!11*!512)
+ sector*!512
+ offset

b) Adresse = Start des Diskpuffers + sector*!512 + offset

Im folgenden illustrierenden Beispiel wird der Diskettenname der Disk im internen Laufwerk ausgegeben:

CD 0:

internes Laufwerk anwählen, falls noch nicht geschehen.

RT !70 !20

lädt !20 Tracks beginnend mit dem Track !70 von der Disk im internen Laufwerk in den Diskkettenpuffer

N T!80S00!433

berechnet automatisch die Adresse des Diskettennamens im Speicher und zeigt diesen an (siehe auch N-Befehl)

5.5.2 WT - Befehl Tracks schreiben

Syntax: WT strack num source

schreibt auf die Diskette im aktuellen Laufwerk beginnend ab dem Track strack (0-!159) num Tracks. Die Daten, die geschrieben werden sollen stehen hierbei ab der Adresse source. Hiermit weisen wir nochmals darauf hin, daß anstelle jeder Adresse, also auch dieser, falls mit einen Diskettenpuffer gearbeitet wird (RT-Befehl ohne dest Angabe), auch das spezielle track/sektor/offset format verwendet werden kann.

Bsp.: Einlesen und wieder Abspeichern von zwei Tracks (Track !10 und !11) mit und ohne Diskettenpuffer

RT !10 2
WT !10 2 T!10

das war mit Diskettenpuffer

RT A 2 1000
WT A 2 1000

ohne Diskettenpuffer, die Daten stehen jetzt "für immer" ab $1000.

5.5.3 DMON - Befehl Diskettenpuffer anzeigen

Syntax: DMON

gibt aus, an welcher Position im Speicher sich der gerade verwendete Diskettenpuffer befindet (nach RT-Befehl).

Bsp.: DMON

5.5.4 CLRDMON - Befehl Diskettenpuffer zurücksetzen

Syntax: CLRDMON

löscht sämtliche Daten im Diskettenpuffer, indem der alte Speicherinhalt wieder restauriert wird.

Bsp.: CLRDMON

5.5.5 -CHK - Befehle Berechne von Block-Checksummen

Syntax: a) BOOTCHK address
        b) DATACHK address
        c) BAMCHK address

zu a)
berechnet die BootBlock Checksumme der beide Sektoren, die sich ab der Adresse address befinden und trägt Sie an der richtigen Stelle ein.

zu b)
berechnet die Datenblock Checksumme des Sektors, der sich ab der angegebenen Adresse befindet und trägt Sie an der richtigen Stelle ein.

zu c)
dito mit einem BAM-Sektor

Bsp.: RT 0
      BOOTCHK T0S0

5.6 Maschinensprach-Monitor - Befehle

5.6.1 A - Befehl Direkt Lineassembler

Syntax: A address

ruft den Direkt-Assembler auf. Dabei wird zunächst die angegebene Adresse ausgegeben. Jetzt können Sie den gewünschten Maschinenbefehl in der üblichen Assembler-Schreibweise eingeben. Durch einfaches drücken der ESC-Taste oder der RETURN-Taste ohne Eingabe eines Assembler-Befehls, kann der Direkt Assembler wieder verlassen werden.

Bsp.: A 700000
      ;070000 ADDQ.l #1,D0
      ;070002 RTS
      ;070004

5.6.2 B - Befehle Breakpoints

Syntax: a) B
        b) BS address
        c) BD address
        d) BDA

zu a)
zeigt die derzeitig gesetzten Breakpoints an.

zu b)
setzt einen Breakpoint auf die angegebene Adresse. Falls kein Breakpoint mehr eingefügt werden kann, wird die Fehlermeldung "NO FREE ENTRY!" ausgegeben.
Es sind dabei maximal 5 Breakpoints möglich!

zu c)
löscht den Breakpoint auf der angegebenen Adresse. Falls dieser nicht gesetzt war, wird die Fehlermeldung "BREAKPOINT NOT FOUND" ausgegeben.

zu d)
löscht alle gesetzten Breakpoints.

Die Breakpoints werden erst beim Verlassen des Moduls aktiviert, wobei Breakpoints, die auf den aktuellen PC gesetzt wurden nicht aktiviert werden, da das Modul sonst sofort wieder aufgerufen werden würde.
Arbeitet das laufende Programm einen gesetzten Breakpoint ab, wird das Programm automatisch unterbrochen und der Modul-Editor aktiviert sich. Außerdem wird die Meldung "BREAKPOINT RAISED AT" und die Adresse des Breakpoints ausgegeben.

WAS IST EIN BREAKPOINT?
Ein Breakpoint (dt. Stopppunkt) ist ein Befehl. Wenn der Prozessor diesen Befehl abgearbeitet hat, wird das Programm

unterbrochen und ein anderes Programm, im diesem Fall, der Modul-Editor aufgerufen.
Anwendung: Man ist in der Lage das laufende Programm gezielt an einer ganz bestimmten Stelle abzubrechen und mit dem Modul-Editor zu analysieren.

```
Bsp.: BS 4348
      BREAKPOINT INSERTED
      B
      BREAKPOINTS:
      004348
```

5.6.3 MW - Befehle Memwatchpoints

```
Syntax: a) MW
        b) MS address
        c) MD address
        d) MDA
```

zu a)
Zeigt alle im Moment benutzten Memwatchpoints an.

zu b)
setzt einen Memwatchpoint an die angegebene Adresse.

zu c)
löscht den Memwatchpoint bei der angegebenen Adresse.

zu d)
löscht alle Memwatchpoints.

Verändert das laufende Programm einen Memwatchpoint, d.h. das laufende Programm hat den Inhalt der mit address angegebene Adresse verändert, wird das Programm automatisch unterbrochen und der Modul-Editor aktiviert sich. Außerdem wird die Meldung "MEMORYBYTE xxxxxx HAS CHANGED FROM xx TO xx" ausgegeben. Durch die Memwatchpoints wird das laufende Programm stark abgebremst was aber normal ist, da regelmäßig die mit MS angegebenen Adressen überprüft werden müssen. Bei entfernen aller Memwatchpoints mit z.B. "MDA" läuft das laufende Programm aber wieder mit normaler Geschwindigkeit.

**ACHTUNG: Da die Prozessorhardware es nicht zuläßt Interrupt-routinen zu tracen, werden Veränderungen, die vom einem Interrupt aus erfolgen, erst nach Beendigung des Interrupt-Programms festgestellt.
Falls Sie Befehle benutzen, die die Trace-Funktion des Prozessors ausnutzen (Memwatchpoints,TR/ST-Befehl), so dürfen folgende Funktionen nicht in Betrieb sein: Bootblockcoder, Bootselektor, SETEXCEPT-Befehl, Breakpoints.**

```
Bsp.: MS 4348
      MEMWATCHPOINT SET
      MW
```



```
      LIST OF MEMWATCHPOINTS: 000438
```

5.6.4 ST/TR - Befehl Programmeinzelschrittbetrieb

```
Syntax: a) TR (num)
        b) ST (num)
```

zu a)
Verläßt das Modul und kehrt nach dem Abarbeiten von num Maschinenbefehlen wieder in den Modul-Editor zurück, wobei automatisch eine Registerausgabe erfolgt. Dabei werden Unterprogrammaufrufe mit der JSR und BSR Instruktion als ein Maschinenbefehl gezählt.
Dieser Befehl arbeitet nur, wenn keine Memwatchpoints gesetzt sind!

zu b)
Verläßt das Modul und kehrt nach dem Abarbeiten von num Maschinenbefehlen wieder in den Modul-Editor zurück, wobei automatisch eine Registerausgabe erfolgt.
Dieser Befehl arbeitet nur, wenn keine Memwatchpoints gesetzt sind!

Beachten Sie den Hinweis bei den Memwatch-Befehlen.

```
Bsp.: D0
      -0000100 NOP
      -0000102 NOP
      -0000104 JSR 200
      -0000106 NOP
      TR 3
Modul wird verlassen und automatisch wieder aktiviert, dann
      D0
      -0000106 NOP
```

5.6.5 C - Befehl Copper Ass-/Disassembler

```
Syntax: C 1|2|address
```

disassembliert die Copper-Liste ab der angegebenen Adresse. Falls anstelle der Adresse die Ziffer 1 oder 2 angegeben wurde, wird die Adresse der ersten bzw. zweiten aktuellen Copper-Liste verwendet. Das daraus resultierende Listing wird direkt vom Copper-Assembler verarbeitet, sodaß Sie direkt in der Copperliste editieren können. (RETURN in jeder Zeile nicht vergessen!)

WAS IST DER COPPER?
Der Copper ist ein Amiga-Spezial-Chip welcher wie ein Prozessor sein eigenes Programm abarbeitet. Im grunde genommen kann der Copper nur festgelegte Daten in die Chipregister

schreiben (ab der Adresse $dff000). Da er dies aber an genau festgelegten BildschirmStrahl-Positionen machen kann, lassen sich erstaunliche Effekte erzielen. Der Copper wird normalerweise für den Bildschirmaufbau verwendet.

WAS IST EINE COPPER-LISTE
Die Copper-Liste ist nichts anderes, als ein Programm für den Copper.

Syntax der Copper-Befehle

```
      MOVE #daten,adr
        daten := zu schreibende Daten
        adr := Offset des zu beschreibenden Chipregisters
Kurz: M #daten,adr
      WAIT (x,y,xmask,ymask,bfd)
        x := vertikale Strahlposition
        y := horizontale Strahlposition
        xmask := vertikales Maske
        ymask := horizontale Maske
        bfd := Blitter-finished-disable-Bit
Kurz: W (x,y
      SKIP (x,y,xmask,ymask,bfd)
        parameter siehe WAIT-Befehl
Kurz: S (x,y
```

Bsp.: Ändern der Hintergrundfarbe der Workbench

```
      C 2
      ~00004436 WAIT ($0,$2F,$1FC,$7F,$1)
      ~0000443A MOVE #$005A,$180
                      ^^^
                    ändern!
```

Nach dem verlassen des Moduls wird man die geänderte Hintergrundfarbe bewundern können.

5.6.6 COMP - Befehl Speicherbereiche vergleichen

Syntax: COMP start end dest
  Kurz: V start end dest

vergleicht den Speicherbereich von start bis end mit dem Speicherbereich ab dest und gibt Unterschiede anhand ihrer Adressen im dest - Bereich aus.

```
Bsp.: M 100
      :000100 12 11 01 12 12 01 00 00 ........
      :COMP 100 103 103
      000104
      READY.
```

5.6.7 D - Befehl Disassembler

Syntax: D (0|address)

disassembliert ein M68000 Maschinenprogramm ab der angegebenen Adresse. Falls keine Adresse angegeben wurde, wird ab der zuletzt disassemblierten Zeile (voriger D-Befehl) weitergemacht. Sollte der erste vorkommende D - Befehl ohne Adresse eingegeben werden, wird als Adresse die genommen, bei der das Programm beim Verlassen des Moduls fortgesetzt wird. Gibt man anstelle der Adresse die Ziffer 0 an, wird für die Adresse der Program-Counter eingesetzt.

```
Bsp.: D 0
      -01234 SUBQ.W #1,$000500
```

5.6.8 E - Befehl Chipregister editieren

Syntax: E (registeroffset)

zeigt den Inhalt des mit registeroffset angegebenen Custom-Chip Registers hexadezimal und binär an. Wird kein Register angegeben, wird das Register 0 ausgegeben. Der Inhalt dieses Registers kann nun durch einfaches Überschreiben (plus <RETURN>) geändert werden. Der geänderte Inhalt wird jedoch nicht direkt, sondern erst beim Verlassen des Moduls gesetzt.
Wie Sie vielleicht schon bemerkt haben stehen ja zwei gleiche Werte (Hexadezimal und Binär) hinter der Chipregisternummer. Wenn Sie nun das ChipRegister verändern wollen, dürfen Sie nur den Wert in EINEM der zwei Werte verändern z.B. den Hexadezimalen.
Man beachte, daß JEDES Chipregister sowohl gelesen, als auch beschrieben werden kann. Da dies physikalisch nicht möglich ist, wird dies vom E-Befehl logisch korrigiert und führt so dennoch in jedem Fall zu einem richtigen Ergebnis. Schreibt man zum Beispiel auf das Nur-Lese-Register DMACONR (Register $2), so wird dies automatisch auf das entsprechende Schreibregister umgeleitet, ließt man das Nur-Schreib-Register COLOR00 aus (Hintergrundfarbe), so erhält man den zuletzt von dem PROZESSOR (!) in das Chipregister geschriebenenen Wert.
Oft kommt es vor, daß Werte, die man von Hand in die Chipregister einträgt bei jedem Bildaufbau (50 mal pro Sekunde) vom Copper überschrieben werden, so daß "von Hand" keine sichtbaren Wirkungen auftreten. So werden z.B. die Farben des aktuellen Bildes oft vom Copper in die jeweiligen Farbregister geschrieben, so daß Änderungen der Farbregister mit Hilfe des E-Befehls keine Wirkung zeigen, da Sie vom Copper ja wieder überschrieben werden! Wollen Sie in einem solchen Fall die Farben aber trotzdem ändern, müssen Sie das Copperprogramm (C-Befehl) entsprechend abändern.

```
Bsp.: E 9A
      ;09A 6302 %0110011100000010
                          ^
```

auf 0 setzen!
(dies sperrt sämtliche Interrupts)
nach Verlassen des Moduls (X-Befehl) sind sämtliche Interrupts gesperrt, d.h. zum Beispiel, daß sich der Mauszeiger nicht mehr rührt u.s.w.

Um nun aus diesem mißlichen Zustand wieder herauszukommen drücken Sie den Modul-Taster und geben Sie folgende Befehle ein

```
E 9A
;09A 4302 %0010001100000010
          ^
```

auf 1 setzen !
(dies erlaubt wieder alle Interrupts!)

5.6.9 F - Befehle Such-Befehle

Syntax: a) F string(,start end)
b) FS string(,start end)
c) FR string(,start end)
d) FA address(,start end)
e) FAQ address(,start end)

zu a)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von start bis end nach dem angegebenen String und gibt die gefundenen Adressen der Reihe nach aus.

zu b)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von start bis end nach dem angegebenen String, macht aber keine Unterscheidung zwischen Groß- und Kleinbuchstaben (dadurch besonders geeignet ASCII-Texte zu finden)

zu c)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von start bis end nach dem angegebenen String, jedoch relativ, D.h. wird beispielsweise der String 0A 03 0B angegeben, sucht er nach folgender Bytefolge: xx xx-7 xx+1, wobei xx beliebig ist! So findet der FR-Befehl auch die Bytefolge 15 0E 16 oder 38 31 39.
Außerdem wird nach der Adresse des gefundnen Strings noch der Offset ausgegeben, der die Differenz zwischen dem angegebenen und dem gefundenen String angibt.
Der FR-Befehl findet so z.B. dann Anwendung, wenn nach einem Text gesucht werden soll, der in einem anderen Zeichencode, als dem ASCII-Code im Speicher steht, die Buchstaben aber dennoch, wie meist, in aufsteigender Reihe geordnet sind.

zu d)
durchsucht den gesamten Speicher, oder falls angegeben den Speicherbereich von start bis end nach Maschinenbefehlen, die in irgendeiner Weise auf die angegebene Adresse zugreifen. Gefunden werden die Befehle, die mit folgenden Adressierungsarten auf die angegebene Adresse zugreifen:
- absolut kurz/lang
- Adressregister indirekt mit/ohne Adressdistanz
- PC-relative Adressierung mit Adressdistanz

zu e)
wirkt wie der FA-Befehl, nur ist der FAQ-Befehl ca. doppelt so schnell. Der FAQ-Befehl ist allerdings nicht so sicher wie der FA-Befehl, d.h. wenn der FAQ-Befehl nichts findet, so kann der FA-Befehl erfolgreicher sein!

ACHTUNG: Damit der FA/FAQ-Befehl alle gewünschten Zugriffe korrekt finden kann, muß der augenblicklich auf dem AMIGA laufende Task das zu untersuchende Programm sein (bei Spielen ist er das i.A. automatisch). Den Namen des augenblicklich laufenden Tasks erfährt man mit dem TASKS-Befehl (siehe dort).

Bsp.: F "ABC",100 5000
SEARCH FROM 000100 TO 005000
00000480
READY.
M 480
:00000480 41 42 43 2A 45 46 47 2A ABC*EFG*
FR "ABC"
SEARCH FROM 000000 TO 080000
0000480 OFFSET: 00
0000484 OFFSET: FC
READY.
FA 100 2200
SEARCH FROM 000000 TO 005000
00004024 ANDI.W #$1234,$00000100
READY.

5.6.10 G - Befehl Programm an Adresse fortsetzen

Syntax: G (address)

setzt das unterbrochene Programm an der angegebenen Adresse fort. Wird keine Adresse angegeben, wird der aktuelle PC verwendet, d.h. das Modul wird normal verlassen.
Dieser Befehl solte nur mit äußerster Vorsicht benutzt werden, da bei sinnlosen Adressen sich das unterbrochene Programm meist unwiederbringlich verabschiedet.

Bsp.: G FC00D2

springt in die Kickstart Reset-Routine -> es wird ein SoftReset ausgelöst.

5.6.11 LM - Befehl Speicherbereich einladen

Syntax: LM (path)name, dest

lädt den unter dem Namen name im Unterverzeichnis path oder dem aktuellen Unterverzeichnis abgespeicherten Speicherbereich (normale Datei) ab der angegebenen Adresse dest in den Speicher.

Bsp.: LM "MEM",100

5.6.12 SM - Befehl Speicherbereich abspeichern

Syntax: a) SM (path)name, start end
b) SMDATA (path)name, start end
c) SMDC (path)name, start end

zu a)
speichert den Speicherbereich von start bis end unter dem Namen name im aktuellen oder angegebenen Verzeichnis auf Diskette ab. Beachten Sie, daß der zu speichernde Speicherbereich "am Stück" im Speicher (RAM/ROM) des Amiga liegen muß und eine Länge von ca. 800Kb (bei einer leeren Diskette) nicht überschreiten darf.
Das erzeugte File enthält danach den Speicher direkt als Bytefolge (Binärfile).

zu b)
gleiche Funktion wie SM-Befehl, jedoch wird ein File erzeugt, das den angegebenen Speicherbereich in Form von BASIC DATA-Zeilen enthält. So gespeicherte Speicherbereiche können dann einfach in Basic Programme eingebunden (Merge-Befehl im BASIC) werden.

zu c)
gleiche Funktion wie SMDATA-Befehl, jedoch enthält das erzeugte File DC-Zeilen, wie sie von jedem Assembler verstanden werden und kann dadurch (mit Include-Anweisung des Assemblers) direkt in einem Assembler Source-File verwendet werden.

Bsp.: SM "MEM", 100 1000

5.6.13 M - Befehl Zeige und editiere Speicher

Syntax: M address

gibt den Speicher ab der angegebenen Adresse aus. Im ausgegebenen Listing kann dann direkt (nur im Hex-Bereich) editiert werden. (Nach geänderter Zeile <RETURN> nicht vergessen!)

```
Bsp.: M 234
      :00000234 11 20 2a 00 10 ff fe fd . *.....
```


5.6.14 MEMCODE - Befehl Speicherbereich kodieren

Syntax: MEMCODE start end code

kodiert den Speicherbereich von start bis end mit der Kodezahl code (0-$ff). So kodierter Speicher kann wieder decodiert werden, indem man ihn nochmals mit derselben Kodezahl kodiert.

```
Bsp.: M 100
      :00000100 40 41 42 43 44 45 46 47 ABCDEFGH
      MEMCODE 100 108 1
      M 100
      :00000100 41 40 43 42 45 44 47 46 BADCFEHG
      MEMCODE 100 108 1
      M 100
      :00000100 40 41 42 43 44 45 46 47 ABCDEFGH
```

5.6.15 ADD - Befehl Wert zu Speicherbereich addieren

Syntax: ADD start end value

Addiert den Wert value byteweise zu den Bytes des angegebenen Speicherbereichs.

```
Bsp.: M 100
      :00000100 40 41 42 43 44 45 46 47 ABCDEFGH
      ADD 100 108 3
      M 100
      :00000100 43 44 45 46 47 48 49 4a DEFGHIJK
```

5.6.16 N - Befehl Speicher als Text ausgeben

Syntax: a) N address
b) NQ address
c) NO (offset)

zu a)
gibt den Speicher ab der angegebenen Adresse als Text aus. Der Text wird gewonnen, indem zu jedem Speicherbyte ein interner Offset (Konstante), der mit dem NO-Befehl vorgewählt werden kann, hinzuaddiert wird. Das Ergebnis wird dann als ASCII-Code interpretiert. Falls nichts anderes angegeben wurde, wird diese Konstante als 0 angenommen, d.h. normalerweise wird der Speicher direkt als ASCII-Text interpretiert. Es kann direkt im ASCII-Text editiert werden.

zu b)
gibt den Speicher ab der angegebnen Adresse als Text aus. Es werden aber alle nichtdruckbaren Zeichen ausgelassen.
Die Ausgabe kann wie gewöhnlich mit <SHIFT> gestoppt und mit <ESC> abgebrochen werden. Am Ende des NQ-Befehls wird die Adresse des zuletzt untersuchten Speicherbereichs ausgegeben.

zu c)
setzt den Offset beim N-Befehl auf den angegebenen Wert, oder gibt den aktuellen Offset aus, falls kein Parameter angegeben wird.
Hier kann z.B. auch der Wert Verwendung finden, der beim FR-Befehl als Offset ausgegeben wurde, um auch andere Texte, als ASCII-kodierte lesen zu können.

```
Bsp.: M 1234
      :001234 41 42 43 44 00 00 00 00 ABCD....
      N 1234
      ;001234 ABCD............................
      NO 1
      N 1234
      ;001234 BCDE............................
```

5.6.17 O - Befehl Speicherbereich füllen

Syntax: O string, start end

füllt den angegebenen Speicherbereich von start bis end mit dem angegebenen String. Falls der String kleiner ist wie der zu füllende Speicherbereich, wird er wiederholt in den Speicher geschreiben.

```
Bsp.: O "AMIGA", 0 80000
     N 0
      ;000000 AMIGAAMIGAAMIGAAMIGAAMIGAAMIGAAM
     O 0, 0 4
      M 0
      :000000 00 00 00 00 41 41 4D 49 ....AAMI
```

5.6.18 R - Befehl CPU Register zeigen/editieren

Syntax: R (register value)

setzt, falls angegeben, das CPU Register auf den angegebenen Wert und gibt sämtliche Register auf dem Bildschirm aus.
Für register schreibt man:
- Datenregister: D0, D1, D2, ..., D7
- Adressregister: A0, A1, ... A7
- User Stackpointer: SP
- Statusregister: SR
- Programcounter: PC

- Flags im SR: FT, FS, FV, FC, FZ oder FX
- Interruptmaske: FI

Bsp.: R PC FC00D2

setzt PC auf die Reset Routine des Kickstarts

5.6.19 SETEXCEPT - Befehl Exceptionhandler installieren

Syntax: SETEXCEPT
  Kurz: SEXC

installiert den Exception Handler des Moduls, der bewirkt, daß bei den meisten Systemabstürzen (siehe unten) zunächst kein GURU ausgelöst wird, sondern das Modul aufgerufen wird. Es meldet sich dabei mit der Meldung, welche Exception den GURU verursacht hat, und wo etwa der Fehlerhafte Befehl steckt.
Abgefangen werden folgende Exceptions:
- Adressfehler
- nichtcodierter OP-Code
- Division durch null
- Line A Emulator
- Line F Emulator

5.6.20 TRANS - Befehl Speicherbereich kopieren

Syntax: TRANS start end dest
  Kurz: I start end dest

kopiert den Speicherbereich von start bis end in den Speicher des Amiga ab der Adresse dest.

Bsp.: TRANS 100 200 1000

kopiert $100-$1FF (=$100 Bytes) nach $1000-$10FF

5.6.21 W - Befehl CIA - Register anzeigen/editieren

Syntax: W (register)

stellt den Inhalt des angegebenen Registers der beiden CIA's auf dem Bildschirm dar. Die Inhalte der Register können direkt im Listing verändert werden. Wird keine Registernummer angegeben, wird das CIA Registerpaar $0 ausgegeben.
Der erste binär dargestellte Wert ist stets der Inhalt des CIA-A, der zweite der Inhalt von CIA-B.
Es werden dabei mehr als die physikalisch vorhandenen 16 Register pro CIA ausgegeben. Die Werte ab "Register" 16 sind

CIA-interne Zustände, die auf diese Weise vom Modul gespeichert werden und nicht verändert werden sollten!

WAS IST EIN CIA?
Bei den CIA's ((C)omplex (I)nterface (A)dapter) handelt es sich um zwei 8520 IC's. Die CIA's haben z.B. die Aufgabe einige Diskettensignale zu verwalten und besitzen unter anderem noch mehrere frei programmierbare Timer.

```
Bsp.: W
      '00 %11111100 %11111111

            auf 1 setzen!
```

verläßt man nun das Modul (X-Befehl) geht die Power LED aus!

5.6.22 WS - Befehl String in Speicher schreiben

Syntax: WS string,address

legt den String string an der Adresse address ab.

```
Bsp.: M 100
      :00000100 00 00 00 00 00 00 00 00 ........
      WS "ABC",100
      M 100
      :00000100 41 42 43 00 00 00 00 00 ABC.....
```

5.6.23 X - Befehl Unterbrochenes Programm fortsetzen

Syntax: X

verläßt den Modul-Editor und setzt das unterbrochene Programm and der Stelle fort, an der es unterbrochen wurde (entspricht dem G-Befehl ohne Adresse).
Achten Sie darauf, daß eventuell vom Programm benötigte Disketten, wie zum Zeitpunkt des Abbruchs oder des Freezens wieder eingelegt sind.

Bsp.: X

5.6.24 Y - Befehl Speicherbereich binär anzeigen/editieren

Syntax: a) Y address
        b) YS (bytes)

zu a)
gibt den Speicher ab der angegebenen Adresse binär aus und zwar soviele Bytes, wie mit Hilfe des YS-Befehls eingestellt

worden sind.
Es kann direkt im Listing editiert werden. (<RETURN> nicht vergessen!)

zu b)
setzt die Byteanzahl bytes für den Y-Befehl auf den angegebenen Wert. Es können Werte zwischen 1 und 8 angegeben werden.
Wird keine Byteanzahl angegeben, wird der augenblicklich eingestellte Wert ausgegeben. Es werden allerdings nicht die Bytes ausgegeben, sondern die Anzahl der Bytes in Bits (1 Byte = 8 Bit)

```
Bsp.: Y 100
      .000100 %0011001100110011
      YS 1
      Y 100
      .000100 %00110011
      .000101 %00110011
      YS
      CURRENT BIT WIDTH: !08
```

5.6.25 ? - Befehl Mini-Taschenrechner

Syntax: ? (-)value (+|-|*|/ value) ...

berechnet den angegebenne Term OHNE Berücksichtigung der "Punkt vor Strich"-Regel und gibt das Ergebnis Binär, Dezimal und Hexadezimal aus.

```
Bsp.: ? !10+4*A
      %00000000000000000000000000110010 = $32 = !50
```

5.7 Befehle um Betriebssystem-Parameter zu untersuchen

Naturgemäß funktionieren alle folgenden Befehle mit Außnahme von INFO und EXCEPTIONS nur dann richtig, wenn das unterbrochene Programm das Betriebssystem auch nutzt, was bei Spielen nicht immer der Fall ist. Sollte das Betriebssystem nicht benutzt werden, d.h. sind die Betriebssystem Parameter erkennbar zerstört, wird die Meldung "EXECBASE NOT VALID" ausgegeben.

5.7.1 INFO - Befehl Amigastatus und einige Chipregs zeigen

Syntax: INFO (picnr)

stellt die wichtigsten System-Parameter des Amiga, die auch den Bildaufbau wesentlich bestimmen, dar. Bei manchen Bildern kommt es vor, daß das Bildschirmbild sich aus mehreren

Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Die Systemparameter können sich nur auf einzelne Bilder beziehen. Welches Bild man nun meint, gibt man mit picnr an.
Läßt man die Bildnummer weg, wird das erste Bild analysiert. U.a. werden die Farben und Adressen der Bitplanes des aktuellen Screens angezeigt, aber auch Sprite- und Audiodaten und die aktuellen Positionen der Floppy-Schreib/Lese-Köpfe. Um deren Position zu bestimmen, muß übrigens der LeseKopf jedes Laufwerks bewegt werden, was man deutlich hören kann, was einer eventuell eingelegten Diskette aber auf keinen Fall schadet!

Die Ausgabe des INFO-Befehls:
- einige Custom-ChipRegister deren Bedeutung in der Fachliteratur nachzulesen ist.
- TDF0, TDF1, TDF2, TDF3 geben die aktuelle Lesekopfposition in Tracks an. "****" bedeutet, daß dieses Laufwerk nicht vorhanden ist.

5.7.2 DEVICES - Befehl Devices des Systems ausgeben

Syntax: DEVICES
  Kurz: DEV

5.7.3 INTERRUPTS - Befehl Interrupts des Systems ausgeben

Syntax: INTERRUPTS
  Kurz: INT

Ausgegeben werden nur ordentlich dem Betriebssystem "mitgeteilte" Interrupts. Um die Adressen der tatsächlicen Interruptroutinen zu erfahren, müssen Sie den EXCEPTIONS-Befehl verwenden.

5.7.4 LIBRARIES - Befehl System Libraries ausgeben

Syntax: LIBRARIES
  Kurz: LIB

5.7.5 PORTS - Befehl Ports des Systems ausgeben

Syntax: PORTS

5.7.6 RESOURCES - Befehl Resources des Systems ausgeben

Syntax: RESOURCES
  Kurz: RES

5.7.7 TASKS - Befehl Laufende Tasks ausgeben

Syntax: TASKS

die Tasks werden dabei in drei Kategorien eingeteilt: Running, Ready und Waiting Task. Das Programm, das im Moment des Programmabbruchs bearbeitet wurde, steht unter dem Eintrag Running Task. Die Adressen vor den Tasks geben die Basisaddresse der zugehörigen Task-Struktur an (s. weiterführende Literatur).

5.7.8. AVAIL - Befehl Freien Speicher ausgeben

Syntax: AVAIL

5.7.9 CHIPREGS - Befehl Chip-Register Namen ausgeben

Syntax: CHIPREG
  Kurz: CREG

gibt eine Liste der im Amiga verwendeten Custom-Chip-Register mit Namen und Offsets aus.

5.7.10 EXCEPTIONS - Befehl Exceptions/Interrupts ausgeben

Syntax: EXCEPTIONS
  Kurz: EXC

gibt die augenblickliche Belegung folgender (Prozessor)-Sprungvektoren aus: Exceptions, Interrupts, Traps

5.7.11 EXECBASE - Befehl Execbase ausgeben

Syntax: EXECBASE
  Kurz: EXEC

gibt Namen, Offsets und augenblickliche Inhalte sämtlicher

ExecBase Einträge aus.

5.7.12 DCHIP - Befehl Chipregister/Beschreibung ausgeben

Syntax: DCHIP chipregname

Gibt eine Beschreibung des angegebenen Chipregisters aus.

Bsp.: DCHIP intreq

5.7.13 ALERT - Befehl Guru-Beschreibung

Syntax: ALERT (gurunr)

Gibt eine Liste aller Gurunummern mit Beschreibung aus, falls keine Gurunummer angegeben ist.

5.7.14 ASCII - Befehl ASCII-Tabelle ausgeben

Syntax: ASCII

Gibt eine ASCII-Tabelle aus.

5.8 Verschiedene Befehle

5.8.1 MEGASTICK - Befehl Joystick-Key-Simulation setzen

Syntax: MEGASTICK (1)
  Kurz: MST (1)

Mit dem Befehl MEGASTICK ist es möglich im laufenden Programm durch Joystick-Bewegungen Tastendrücke zu simulieren. Anwendungen sehen z.B. wie folgt aus:
- In einem Actionspiel für Joystick ist der Feuerknopf zum Feuern da. Gibt es aber noch weitere Funktionen, wie z.B. Hyperbomben, so werden diese oft mit einer Taste (z.B. SPACE) ausgelöst. Stellen Sie nun auf Modul-Dauerfeuer (mit Preferences) und programieren Sie den Spielstick mit dem MEGASTICK-Befehl so, daß auf Feuerknopfdruck ein Tastendruck auf SPACE simuliert wird (s.u.). Ergebnis: Dauerfeuer und Hyperbomben per Joystick!!!
- Ein anderes Spiel (z.B. oft PD - Spiele) steuert die Spielfigur nur über Tastatur, dann kann man den Joystick so programmieren, daß wenn z.B. der Joystick nach oben gedrückt wird, die Taste für Spielfigur nach oben simu-

liert wird, und schon kann man mit dem Joystick spielen. Am besten kann man die Simulation anwenden, wenn man im Besitz eines Joysticks ist, der zwei getrennte Feuerknöpfe (wie die Maus) besitzt. Dann kann man den zweiten Feuerknopf in Kombination mit Joystickbewegungen frei programmieren, ohne beim Spielen mit den normalen Funktionen in Konflikt zu kommen.

Programmieren des/der Joysticks:
Falls nur der Joystick von Player 1 programmiert werden soll, muß der Wert 1 mit angegeben werden!
Nach Eingabe des Befehls MEGASTICK (oder MST) gelangt man in das Programmier-Menue.
Will man nun z.B., daß durch die Kombination Joystick nach oben + Knopf0 gedrückt die Taste SPACE simuliert werden soll, so verfährt man wie folgt:
Gleichzeitig den Joystick nach oben und den Knopf0 gedrückt halten. Danach (während man den Joystick immer noch betätigt) die SPACE-Taste betätigen und wieder loslassen. Jetzt kann man den Joystick und den Knopf0 wieder loslassen; diese Funktion ist jetzt programmiert und in der Spalte PLAYER1 und Zeile FIRE0 erscheint oben der Tastencode $7f um anzuzeigen, daß diese Bewegung programmiert ist.
Um den letzten Tastencode zu löschen, drückt man bei losgelassenem Joystick eine beliebige Taste (außer ESC).
Um das Programmier-Menue zu verlassen drückt man die ESC-Taste, wodurch die Joystick Simulation auch aktiviert wird (nach Verlassen des Moduls).

5.8.2 NOSTICK- Befehl Joystick-Key-Simulation ausschalten

Syntax: NOSTICK
  Kurz: NST

schaltet die Joystick-Key-Simulation wieder aus.

5.8.3 CLRSTICK - Befehl Joystick-Key-Simulation löschen

Syntax: CLRSTICK
  Kurz: CST

löscht die Programmierung des/der Joystick(s) und schaltet die Simulation aus.

5.8.4 SSTICK - Befehl Joystick-Key-Simulation abspeichern

Syntax: SSTICK (path)name
  Kurz: SST (path)name

speichert die aktuelle Joystick-Key-Simulation unter dem Namen name im Verzeichnis path auf Diskette ab.

5.8.5 LSTICK - Befehl Joystick-Key-Simulation einladen

Syntax: LSTICK (path)name
  Kurz: LST (path)name

lädt eine abgespeicherte Joystick-Key-Simulation mit dem Namen name aus dem Verzeichnis paht wieder ein und aktiviert sie.

5.8.6 VERSION - Befehl Versionsnummer/datum ausgeben

Syntax: VERSION

5.8.7 RAMTEST - Befehl Speicherbausteine des Amiga testen

Syntax: RAMTEST start end

testet den angegebenen Speicherbereich von start bis end (muß am Stück liegen) und gibt fehlerhafte Bytes aus.

5.8.8 PACK - Befehl Speicherbereich komprimieren

Syntax: PACK start end dest crate

komprimiert den Speicherbereich von start bis end und schreibt den komprimierten Speicher ab der Adresse dest wieder in den Speicher.
Mit crate (0-$7fff) wird die Effizienz des Komprimiervorgangs festgelegt. Normale Werte bewegen sich um $20 herum. Je größer crate, desto besser wird komprimiert (dauert aber i.A. länger).
Nach dem Komprimiervorgang wird die Länge der komprimierten Daten ausgegeben. (Zur Verwendung beim Abspeichern und Entkomprimieren)

Bsp.: PACK 70000 70100 71000 30
      000044
packt den Bereich von $70000-$70ff nach $71000
      UNPACK 60000 71044
entpackt den oben komprimierten Bereich wieder, so daß jetzt von $60000 bis $60100 das Gleiche steht wie ab $70000

5.8.9 UNPACK - Befehl Speicherbereiche entfalten

Syntax: UNPACK dest endofpacked

macht den Vorgang des PACK-Befehls wieder rückgängig, indem der komprimierte Speicherbereich wieder ent-komprimiert nach dest geschrieben wird. endofpacked muß die Adresse des letzten gepackten Bytes +1 enthalten.

Bsp.: s. PACK-Befehl

5.8.10 COLOR - Befehl Modulfarben setzen

Syntax: COLOR (back pen)

setzt die Farben des Moduleditors auf die gewünschten Werte, dabei ist back die Farbe des Hintergrundes und pen die der Zeichen. Falls keine, oder nur ein Parameter angegeben wird, werden lediglich die aktuellen Farbwerte ausgegeben.
Die eingegebenen Zahlen back und pen werden dabei als RGB-Farbwerte interpretiert (z.B. 000 = schwarz, F00 = rot, FFF = weis u.s.w.)

Bsp.: COLOR 000 0F0
      COLOR 05A FFF

5.8.11 RCOLOR - Befehl Modulfarben auf Standardwerte setzen

Syntax: RCOLOR

5.8.12 TM - Befehle Speichermerker

Syntax: a) TM
        b) TMS address
        c) TMD address

zu a)
zeigt die aktuell gesetzten Speichermerker an.

zu b)
setzt einen Speichermerker auf die nachfolgend eingegebenen Werte. Es sind maximal zehn Speichermerker möglich. Die Speichermerker stellen dabei einen Notizblock dar, auf dem alle möglichen in einem Spiel wichtigen Speicherstellen vermerkt werden können, die Sie vielleicht mit dem Trainermaker oder anders herausbekommen haben.

5.8.14 SETMAP - Befehl Keymap-Editor

Syntax: SETMAP
  Kurz: KEY

Ruft den Keymap-Editor auf, mit dem man die Tastaturbelegung des Amiga Betriebssystems ändern kann, so daß z.B. Funktionstasten mit nützlichen Strings belegt werden können u.a.

Nach kurzer Zeit meldet sich das Modul dann mit dem Keymap-Bildschirm, auf dem die Tastatur und vier zusätzliche Felder angeboten werden (Load,Save,Install,Exit).
Um eine Taste zu ändern betätigen Sie diese einfach (zuvor evtl. noch Shift, Alternate oder Control betätigen, die sog. Qualifier) oder wählen Sie mit der Maus an. Danach erscheint ein neues Menue mit folgenden Einträgen:

Code:	Tastaturcode der zu ändernden Taste
Sht,Alt,Ctl:	Qualifier, auf die die Taste bisher reagierte
Repeatable:	Taste besitzt Dauerfunktion
Capsable:	Taste wird von CapsLock-Taste beeinflußt
String:	Taste sendet Zeichenketten
Nop:	Taste reagiert nicht
OK:	geänderte Taste eintragen
Cancel:	Änderung rückgängig machen

rechts wird die augenblickliche Belegung der Taste ausgegeben.
Der CURSOR befindet sich jetzt automatisch auf dem String, den die Taste z.Zt. sendet. Dieser kann jetzt beliebig geändert werden (max. 128 Zeichen), wobei nicht druckbare Zeichen mit Ihrem Hexcode eingegeben werden können, eingerahmt in "<$" und ">". Dabei ist "<$D> gleichbedeutend mit "<r>" (ASCII-Code für RETURN).
Nach Betätigen der RETURN-Taste erscheint der Mauszeiger und Sie können jetzt in den obigen Feldern gewünschte Veränderungen vornehmen (z.B. Repeatable ausschalten u.s.w.)
Um die Veränderungen zu übernehmen das OK-Feld betätigen, sonst das Cancel-Feld.
Um eine solcherart geänderte Tastaturbelegung in das Amiga Betriebssystem zu installieren, betätigen Sie das Install-Feld.
Um eine (sog) Keymap zu laden oder zu speichern, betätigen Sie das Load-Feld bzw. das Save-Feld.
Das Exit-Feld dient natürlich dazu den Keymap-Editor zu verlassen.
WICHTIG: Der Keymap-Editor arbeitet nur richtig, falls die Multitasking-Oberfläche des Amiga aktiv ist. (z.B. beim Arbeiten mit einem CLI-Fenster)

Abgespeicherte Keymaps können auch mit dem Setmap Befehl der Workbench eingeladen werden. Dazu sollte man die Keymap vorher in den entsprechenden Ordner kopieren. (DEVS:Keymaps)

Die Speichermerker werden beim Speichern des Spiels mit dem SA-Befehl mit abgespeichert und automatisch beim Laden mittels LA/LR-Befehl wieder eingeladen!

Der TMS-Befehl verlangt folgende Eingaben:

TYPE	= Typ des Zählers - geben Sie hier den am nächsten passenden Typ an
COUNTERLENGTH	- Geben Sie nun die Länge des Zählers im Speicher an. 1 = Byte, 2 = Wort, 3 = Langwort (i.A. ist es ein Wort)
MAXCOUNT	= Geben Sie hier den größten Wert des Zählers an, bei dem das Programm noch korrekt arbeitet (!127 = $7F hat sich bewährt, da größte Vorzeichenbehaftete Zahl)
PLAYER	- Geben Sie nun die Spielernummer an, bei dem diese Parameter gelten (normalerweise Spieler Nummer eins)

zu c)
löscht den Speichermerker, der auf die angegebene Adresse zeigt. Falls dieser nicht existiert, wird die Meldung "MEMORYCOUNTER NOT FOUND" ausgegeben.

5.8.13 SPR - Befehl Hardware-Sprites editieren

Syntax: SPR nr1|address1 (nr2|address2)

Gibt den Sprite der Nummer nr1 oder den Sprite, der ab der Adresse address1 im Speicher (Chip-RAM!) liegt, als editierbares Listing auf dem Bildschirm aus, wobei die Spritefarben durch die Ziffern 0 bis 3 dargestellt werden.
Wird eine zweite Spritenummer oder -adresse angegeben, wird der Sprite als attached Sprite ausgegeben und seine Farben durch die Ziffern 0 bis 9 und A bis F dargestellt.
In jedem Fall stellt die Farbe 0 die durchsichtige Farbe dar.

Bsp.: von der Workbench aus:

```
Spr 0
   -001480 1111110000000000
   -001482 1222221000000000
   -001484 1333321000000000
   -001486 1333210000000000
   -001488 1333321000000000
   -00148A 1331332100000000
   -00148C 0110133210000000
   -00148E 0000013321000000
   -001490 0000001332100000
   -001492 0000000131000000
   -001494 0000000010000000
   -001496 0000000000000000
```

jetzt kann der Mauszeiger beliebig verändert werden!
(Nach jeder geänderten Zeile RETURN nicht vergessen!)

ACHTUNG: Änderungen der Keymap machen sich erst beim Neustarten von Programmen oder beim Öffnen eines neuen CLI-Fensters bemerkbar. "Alte" Programme benützen weiter die alte Keymap.

Bsp.: Wollen Sie den String "dir df0:"+<RETURN> auf die Taste Control-D legen, gehen Sie folgendermaßen vor:

1. SETMAP eingeben (oder KEY)
2. Ctrl-Taste drücken (und wieder loslassen)
3. D-Taste drücken
4. mit der DEL-Taste den aktuellen String löschen und "dir df0:<r>" gefolgt von RETURN eintragen.
5. OK-Feld betätigen
6. Install-Feld betätigen
7. Modul verlassen mit dem X-Befehl
8. Neues CLI-Fenster öffnen (NEWCLI)
9. Ctrl-D drücken

5.8.15 RESET - Befehl Soft-Reset auslösen

Syntax: RESET

Verlässt das Modul und löst einen Software-Reset aus.

5.8.16 NTSC - Befehl In den ECS-NTSC Modus (60Hz) schalten

Syntax: NTSC

5.8.17 PAL - Befehl In den ECS-PAL Modus (50Hz) schalten

Syntax: PAL

Achtung: NTSC-Befehl und PAL-Befehl funktionieren nur auf einem Amiga mit einem ECS-Chipset (neues Modell)

5.9 Befehle zur Ansteuerung eines Druckers

5.9.1 NORMALCHAR - Befehl Normale Druckerausgabe

Syntax: NORMALCHAR
  Kurz: NCHAR

siehe auch SMALLCHAR-Befehl.

5.9.2 SMALLCHAR - Befehl Auf Kleinschrift umschalten

Syntax: SMALLCHAR
  Kurz: SCHAR

folgende Druckerausgaben werden mit einem verkleinerten Zeichensatz ausgegeben, um Papier zu sparen.
Umschalten auf normale Ausgabe siehe NORMALCHAR.

5.9.3 PRT - Befehl String auf Drucker ausgeben

Syntax: PRT string

Gibt den angegebenen string auf dem Drucker aus.

Bsp.: PRT "Hello World" A

Handhabung des normalen Trainers (T-Befehle)

Das AMIGA ACTION REPLAY CARTRIDGE stellt Ihnen mit dem T-Befehl einen leistungsfähigen 3 Pass Trainer zur Verfügung, der meisten Spiele zu trainen in der Lage ist.

Aber VORSICHT: Die meisten Spiele werden, einmal getraint, schnell langweilig.

Um ein Spiel zu trainen, d.h. Sie mit beliebig vielen Bildschirmleben auszustatten, sind folgende Arbeitsschritte vonnöten:

1.) Das zu trainende Spiel normal einladen und starten
2.) Dem gestarteten Spiel die Anzahl der Bildschirmleben (kurz Leben) entnehmen (z.B. fünf)
3.) Das Modul aktivieren (Taster)
4.) TrainerMode starten mit dem TS-Befehl (im Bsp.: TS !5)
5.) Das Modul verlassen (X-Befehl) und solange spielen, bis sich die Anzahl der Leben verändert hat. Am besten hat sich in der Praxis die sog. Kamikazemethode bewährt: Sämtliche Gegner mitnehmen, ohne auch nur einen einzigen Schuß abzufeuern...
6.) Dem Spiel die Anzahl der jetzt noch zur Verfügung stehenden Leben entnehmen (z.B. vier) und die Cartridge aktivieren (Taster)
7.) Den Trainvorgang fortsetzen mit dem T-Befehl (im Bsp.: T !4)

8a) Falls keine oder sehr viele Adressen ausgegeben werden bei Schritt 5) weitermachen!

8b) Merken Sie sich die wenigen Adressen und setzen Sie noch einmal bei Punkt fünf fort. Wenn Sie Adressen bemerken, welche immer wieder vorkommen, so haben Sie ziemlich sicher die gesuchte Adresse gefunden. Verfahres Sie wie in Punkt (c), nur müssen Sie nun ausprobieren, welche der gefundenen Adressen Sie verändern müssen.

8c) Falls genau eine Adresse übrigbleibt und der gesammte Speicher durchsucht worden ist, so haben Sie mit ziemlicher Sicherheit die Speicherstelle gefunden, mit deren Hilfe das laufende Programm die Leben mitzählt. Falls der Speicher noch nicht ganz durchsucht worden sein sollte, kann es dennoch sein, daß die gefundene Adresse bereits die gesuchte ist. Zur Sicherheit aber sollten Sie ab Punkt 5) nochmal wiederholen und überprüfen, ob die Adresse bestehenbleibt, dann können Sie sicher sein, die richtige Adresse erwischt zu haben.

Wenn Sie mit dieser Prozedur eine verdächtige Speicherstelle ausfindig gemacht haben, können Sie diese erhöhen, um mehr Bildschirmleben zu erhalten (M-Befehl)
z.B.: Die gefundene Adresse lautet 23686. Geben Sie nun "M 23686" ein. Nun werden sechzehn Bytes ab der Adresse 23868 angezeigt. Der Cursor steht nun direkt auf dem Zähler und Sie können nun eine zweistellige Hexadezimal-

zahl eintragen. Geben Sie nun <RETURN> + <ESC> + "X" + <RETURN> ein. Das Modul wird nun verlassen und Sie können sich nun von der Leistung des Trainers überzeugen!
Sie können die Adresse, bei den Speichermerkern eintragen (TM-Befehl) und so zu jedem Programm immer direkt, ohne erneut suchen zu müssen, die jeweilig wichtigen Adressen zur Verfügung haben.

Aber Vorsicht: nicht jedes Programm "verkraftet" beliebig hohe Werte für Inre Leben. Der Wert $7F = !127 hat sich in der Praxis jedoch als relativ absturzsicher bewährt.

Noch eleganter, als den Zähler von Hand ständig zu erhöhen, ist die Methode, den Maschinenbefehl, der die jeweilige Speicherstelle erniedrigt, zu entfernen. Zu diesem Zweck bietet die Cardtridge den TFD-Befehl an. Dieser sucht (und entfernt) wenn möglich diese Befehle und macht Sie damit absolut unsterblich (am Bildschirm).

Assembler Profis können mit dme FA/FAQ-Befehl sich genau informieren, wie, wann und wo auf den Zähler zugegriffen wird!

Auf gleiche Weise kann man alles trainen, was irgendwie abzählbar ist. Z.B.: Sie haben zu Beginn des Spiels drei Hyper-Bomben zur Verfügung. Starten Sie mit "TS !3". Jetzt werfen Sie eine Bombe und fahren mit "T !2" fort usw. Auf diese Weise finden Sie die Speicherstelle, mit der das Programm die Hyperbomben zählt...

Sollten Sie während des Trainvorgangs so viele Leben verlieren, daß das Spiel endet (GAME OVER): Macht nichts! Starten Sie einfach das Spiel erneut und setzten den Trainvorgang wie gewohnt fort, als wäre nichts geschehen, lediglich die Anzahl der Leben hat sich ja erhöht...

ACHTUNG: Falls der Trainer bzw. der Final Train Befehl (TFD) nicht zum erwünschten Ergebnis führen sollte, lesen Sie sich bitte das Kapitel TIPS&TRICKS durch und probieren Sie die dort genannten Variationen aus! Mit etwas Übung können Sie dann fast jedes Programm trainen.

Tips & Tricks

- Schalten Sie, falls möglich, Ihre Speichererweiterungen mit der Preferences-Seite ab.
Sie ersparen sich dann unnötig lange Speicher- und Ladezeiten und unnötigen Diskettenbedarf beim SA/LA/LR-Befehl. Die Zeiten für die Suchbefehle und nicht zuletzt den Trainer werden ebenfalls verkürzt

- Sollten Sie mit resetfesten Programmen arbeiten, müssen Sie das Speicherlöschen und den Virustest in der Preferences-Seite abschalten.

- Wenn die Speicherlöschoption eingeschaltet ist, arbeitet der Packer beim Freezen besser und schneller.

- Es ist ratsam, wegen der grassierenden Virusgefahr den Resetvirustest und das Speicherlöschen beim Reset eingeschaltet zu lassen, so daß sich keine Viren in Ihrem Amiga festsetzen könen. Sollte bei einem Reset ein Virus erkannt werden (am Flackern des Bildschirms beim Reset zu erkennen), sollten Sie Ihre seit dem letzten Reset benutzten Disketten auf Viren Überprüfen!

- Sollte der Trainer einmal versagen, nicht verzagen...:
Geben Sie beim Trainvorgang (TS/T-Befehl) jeweils die Anzahl der Bildschirmleben (o.ä.) plus eins (!) an. Achten Sie darauf, daß Sie den Zahlenwert immer korrekt angegeben haben. (Immer "!" für dezimale Zahlen eingegeben?)

- Bei manchen Zählern kann es vorkommen, daß der Zählerwert im BCD-Zahlenformat im Speicher steht. z.B.:
Auf dem Bildschirm steht der Zählerwert "978". Normalerweise müsten Sie jetzt mit "TS !978" den Trainvorgang starten. Wegen des BCD-Zahlenformates müsses Sie aber statdessen beim T/TS-Befehl das "!" für dezimal weglassen. Der richtige Befehl lautet also "TS 978"!

- Sollte der TF-Befehl einmal nicht fündig werden und die Adresse der Speicherstelle ist ungerade (Endziffer 1, 3, 5, 7, 9, B, D, F), sollten Sie einmal von der Adresse eins abziehen und TF/TFD-Befehl wiederholen!
z.B.: Der Trainer liefert die Adresse $33EF für den Zähler. Der Befehl "TFD 33EF" hatte aber nichts gefunden, dann probieren Sie einmal die Zähleradresse -1 (Im Beispiel "TFD 33EE")

- Bei Zählern, die größer als !255 werden können (z.B. Score), liefert der Trainvorgang, wie auch sonst, die Adresse des letzten Bytes, dh.h. wenn man das oder die vorhergehenden Bytes verändert, können evtl. sehr große Zählerwerte erreicht werden. Das Byte mit der niedrigsten Adresse dieses Zählers sollte jedoch nicht größer als $7F gewählt werden.

- Normalerweise sind alle Playerdaten wie z.B. Leben, Energie und Punkte sehr nahe im Speicher (RAM) aneinander angeordnet. Wenn Sie also z.B. den SCORE gefunden haben müssen Sie nur in diesem Speicherbereich suchen und mit ziemlicher Sicherheit werden Sie nun auch die anderen Playerdaten finden!

- Wie mache ich mir eine Arbeitskopie von einem kopiergeschützten Original?
1) Falls es kein Nachladeprogramm ist laden Sie einfach das Original ein und FREEZEN Sie es einfach mit dem SA-Befehl ->Fertig!!!
2) Es handelt sich um ein Original, welches Daten von der Diskette während dem Spiel nachladt.
- Kopieren Sie Ihr Original mit einem üblichen Kopierprogramm.
- Laden Sie das Original ein.
- Nachdem der Kopierschutz abgefragt wurde FREEZEN Sie es einfach (SA-Befehl) -> fertig.

- Wie lade ich meine Arbeitskopie meines Originals?
1) Falls es kein Nachladeprogramm ist laden Sie einfach das gefreezte Programm mit dem LR-Befehl ein -> fertig.
2) Es handelt sich um eine Arbeitskopie, welche von Diskette nachlädt:
- Laden Sie das gefreezte Programm mit dem LA-Befehl ein.
- Lefen Sie ohne das Modul zu verlassen die Kopie(n) der Originaldiskette(n) in ihre Laufwerke
- Verlassen Sie das Modul mit dem X-Befehl -> fertig.

- Wenn Sie mit dem TF/TFD/FA/FAQ-Befehl arbeiten wollen, sollten Sie darauf achten, daß der gerade laufende Task (siehe UT-Befehl) das Spiel ist!
z.B.: Wollen Sie in einem Spiel die Adresse finden, welche ihren Zähler schändlich behandelt, (ihn vermindert) so sollten Sie darauf achten, daß der aktuelle Task (UT-Befehl) auch das Spiel ist (sofern das Betriebssystem des Amiga nicht sowieso vom Spiel umgangen wird). Mit etwas Erfahrung werden Sie später schnell erkennen, ob der aktuelle Task der gesuchte Task ist.

-Falls ein Programm nicht mit dem Action Replay Mk II Modul arbeitet sollten Sie die erweiterten Funktionen des Action Replay abschalten z.B: Virustest, Dauerfeuer, Bootselector, Diskcoder, Bootblockcoder, Speicherlöschen

-Tips zur sinnvollen Ausnutzung von Speichererweiterungen auf über 1MB für Funktionen des Moduls.

1. **Externe** Speichererweiterungen werden bei eingeschalteter ClearMem Funktion (s.Preferences) nur gelöscht, wenn Sie vom Modul eingebunden werden. D.h. nur dann, wenn das Add-Feld aktiviert ist, und die Speichererweiterung nicht autokonfigurierend ist, oder gemacht wurde (mit Autoconfig. Off im Preferences Menue).

2. Da die meisten Spiele ohne Speichererweiterungen lauffähig sind, sollten Sie den nicht benötigten Speicher mit dem Modul abschalten und so für Funktionen des Moduls zur Verfügung stellen. (s. Preferences bzgl. Abschalten von Speichererweiterungen).

3. Externe Speichererweiterungen stehen sowohl dem Amiga Betriebesystem, als auch dem Modul nach einem Reset erst ab dem Zeitpunkt zur Verfügung, wenn der Computer versucht von Diskette zu booten, d.h. wenn z.B. die WorkBench-Hand erscheint. Funktionen des Moduls, die die externe Speichererweiterung benötigen (z.B. SaveQuick-File befindet sich dort) können also erst danach benutzt werden.

ANHANG C

Bediendung des Deep-Trainers (TD-Befehl)

Das AMIGA ACTION REPLAY stellt Ihnen mit dem TD-Befehl einen neuartigen Trainer zur Verfügung, der Sie in die Lage versetzt, mit geringfügig größerem Aufwand nahezu alle Spiele zu trainen.

Aber VORSICHT: Die meisten Spiele werden, einmal getraint, schnell langweilig.

ACHTUNG: Der Deep-Trainer benötigt für seine Arbeit einen mindestens 512kB-großen freien Speicherbereich, der dem Amiga Betriebssystem nicht bekannt ist (s. auch Kap. Preferences)

Um ein Spiel mit dem Deep-Trainer zu trainen sind i.A. folgende Arbeitschritte von nöten (als Beispiel versuchen wir die Anzahl der Bildschirmleben zu trainen):

1.) Das zu trainende Spiel normal einladen und starten.
2.) Das Modul aktivieren (mit Taster).
3.) Mit dem TDX Befehl einen neuen Deep Train Vorgang initialisieren
4.) Mit dem TDS Befehl den Deep Trainer starten und sich zu diesem Zeitpunkt die Anzahl der Leben notieren. (Im Beispiel fünf)
5.) Mit dem X Befehl das Modul verlassen und ein Leben verlieren.
6.) Das Modul aktivieren und mit dem TDC Befehl den Trainvorgang fortsetzen. Dabei ist zu beachten, daß man den TDC Befehl nur anwenden darf, falls sich die Anzahl der Bildschirmleben wirklich geändert hat. (Im Beispiel ungleich fünf)
7.) Das Modul mit dem X Befehl verlassen und die Anzahl der Bildschirmleben wieder verändern.
8.) Falls die Anzahl der Bildschirmleben ungleich der notierten Zahl (fünf) ist, weiter mit 6.), sonst
9.) Das Modul aktivieren und mit dem TDS Befehl den Deep Train Vorgang fortsetzen. Dabei werden u.U. einige mögliche Adressen ausgegeben.
10a.) Falls keine Adressen gefunden wurden weiter mit 7.)
10b.) Falls sehr viele Adressen gefunden wurden, kann man mit 7.) fortfahren oder mit dem TDD Befehl einen Teil der gefundenen Adressen löschen, von denen Sie glauben, daß dieser Teil des Speichers nicht die gesuchte Adresse beinhaltet (z.B. weil sich dort der Bildschirmspeicher befindet). Danach mit 7.) fortfahren.
10c.) Falls nur eine oder wenige Adressen ausgegeben werden, können Sie versuchen, z.B. mit dem M-Befehl, die Inhalte der ausgegebenen Adressen zu verändern, und das Resultat des "Eingriffs" nach dem X-Befehl zu begutachten. Sollte die richtige Adresse nicht dabei gewesen sein (der gewünschte Effekt tritt nicht ein), weiter mit 7.)

Haben Sie die richtige Adresse einmal ausgemacht, können Sie weiter verfahren, wie im ANHANG A nach 8c) beschrieben.

Da der Deep Trainer für seine Arbeit keine Zahlen benötigt, sondern nur die Unterscheidung gleich/ungleich, kann man mit Ihm auch sehr gut Dinge trainen, die nicht als Zahlenwert auf dem Bildschirm ausgegeben werden (z.B. Energiebalken), von denen man aber einen gewissen Grundzustand reproduzieren kann (voller Energiebalken o.ä.).

Alphabetische Befehlsliste: